7-1-2007

Information flow between tools early in the engineering design process

Kate Nordland

Information Flow Between Tools Early in the Engineering Design Process

by

Kate Elizabeth Nordland

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Mechanical Engineering

Supervised by

Dr. Edward Hensel Department Head, Mechanical Engineering
Department of Mechanical Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
July 2007

Approved By:

Dr. Edward Hensel
Department Head, Mechanical Engineering
Adviser

Dr. Margaret Bailey
Associate Professor, Mechanical Engineering

Dr. Marcos Esterman
Assistant Professor, Industrial & Systems Engineering

Mr. George Slack
Lecturer, Electrical Engineering

Mr. Mark Smith
Director, Product Development

Dr. Steven Day
Department Representative, Mechanical Engineering

Thesis Release Permission Form

Rochester Institute of Technology

Kate Gleason College of Engineering

Information Flow Between Tools Early in the Engineering Design Process

I, Kate Elizabeth Nordland, hereby grant permission to the Wallace Memorial Library reproduce my thesis in whole or part.

Kate Elizabeth Nordland

Date

Acknowledgments

I'd like to acknowledge Eric Leipold for supporting my efforts over the past two years. Always understanding and helpful, I couldn't have made it this far without him. Even on the long days, being with him reminds me that life could be worse.

I'd also like to acknowledge my parents, Lorrie Carlson and Boyd Nordland, for being completely understanding of my wish to go back to school and pursue my masters. When times were tough, I knew that I could always count on both my mom and my dad.

Lastly I'd like to acknowledge Dr. Hensel for being a great mentor and advisor. Our conversations of what the future holds for implementation of this project were always exciting and inspiring. I've learned so much from working with him on this project. From narrowing a topic selection to proof reading three nights a week, his help has been invaluable.

Abstract

Engineering design processes are recognized by industry as critical for successful new product development. With many existing design processes, it is unexpected that there is not a method for translating between processes. FACETS aims to become a translator between various design processes. To accomplish this, information must be entered, sorted, and displayed in several ways to communicate design intent to the user.

A database schema is proposed to store the qualitative information and relationships associated with the early phases of the engineering design process. The proposed database is implemented using a MySQL database. Tools for conducting the needs assessment and concept development are implemented through a series of scripts that provide a user interface to the database. Two case studies test the appropriateness of the database schema and determine areas for further development.

The scripts developed render tools for students in the Kate Gleason College of Engineering Multidisciplinary Senior Design class to facilitate navigating the early design process. These scripts will collect the customer driven needs of the project and provide tools to help translate those needs into proposed design(s) to develop further. Information is intended to flow between tools, phases and design processes. To demonstrate the information flow between customer needs and design concepts, a mapping is graphically rendered.

Conclusions confirm that the database supports the information contained in the case study. Additional scripts are necessary to provide the graphical wrappers to display the information similar to the case studies. Secondary conclusions highlight a lack of literature regarding the connections between engineering specifications and product functions and the apparent inconsistencies within the quality function deployment House of Quality when implicit relationships are displayed. Recommendations for further work are presented.

Contents

List of Figures

List of Tables

Chapter 1

Statement of Work

There is a large body of knowledge available that defines engineering design processes. With many design processes defined, it is unexpected that there is not a method for translating between processes. A long term goal of the FACETS process outlined in Chapter 3 is to become an open source translator between various design processes. To accomplish this, information must be entered, sorted, and displayed in several ways to communicate design intent to the user.

This thesis proposes a web-based tool set, focused on information determined early in the engineering design process. The data created during the early stages of the design process will be stored in a relational database through a series of web-based scripts to help users capture data and create relationships between the data. These scripts will aid the user in ensuring that data created in one facet flows to other facets as needed.

Specifically, this thesis will be focused on developing a relational database designed to provide a framework for collecting and displaying information related to the Needs Assessment and Concept Development facets through EDGE. These tools will initially be geared towards students in the Multidisciplinary Senior Design class at RIT. A background of the design course sequence at RIT is included in Chapter 3. The EDGE collaboration environment and the FACETS design process are both described in detail. The goal of this thesis is to be installed within the EDGE environment to support the design data in FACETS.

The literature review in Chapter 2 will identify several commonly used design processes and tools. The validity of the results of some of these tools has been called into question.

It is not the purpose of this thesis to address the validity of the information provided by the user. The design process review will confirm that the selection of Needs Assessment and Concept Development as the "early" phases of design is valid. The description of the design tools associated with these phases lay the ground work for the scripts provided.

Table structures for a schema are proposed in Section 4.2. The schema structure is intended to be extensible. The schema should fully capture the information collected from the tools as well as the relationships between information.

The table structure is implemented using a MySQL relational database structure through a series of web-based open architecture, open source scripts written in PHP. MySQL and PHP are both open source software packages. The scripts are used for collecting the information from the user and displaying it back to the user. The user interface will meet the requirements of the design tools, but will not be in final graphical user interface (GUI) form. The scripts are described in Section 4.3.

Two case studies will be conducted using examples found in **Engineering Design** by Dym and Little [3], and **The Mechanical Design Process** by Ullman [9]. These case studies, found in Chapter 5, evaluate the capability of the database to capture information presented in various tools in the context of two different processes. In addition to the information presented in the case studies, additional views will showcase the scripts and databases' adaptability to different processes and described tools, while capturing the data and relationships underlying the processes and tools in a compact, robust form.

Chapter 6 presents conclusions which evaluate the table structure and implementation of the scripts in their effectiveness of supporting information flow. Additional observations are offered.

Recommendations for future work are offered in Chapter 7.

In summary, this thesis will

- Determine similar facets early in engineering design process.
- Identify common tools used in the early facets of the design process.

- Propose a table structure to store information from identified tools.
- Implement the table structure through a series of web-based scripts.
- Demonstrate the web-based scripts through case studies.
- Evaluate the table structure and implementation.
- Recommend considerations for future work.

Throughout the thesis, the font style `typewriter` is used to signify the name of database schemas and fields and the font style sans serif is used to signify the name of a script page or variable.

Chapter 2

Literature Review

2.1 Definition of Engineering Design

There is no one universally accepted definition of engineering design. The Accreditation Board of Engineering and Technology, ABET, gives the following definition [10][p 2]:

> "Engineering design is the process of devising a system, component, or process to meet desired needs."

The text **Product Design and Development** by Ulrich and Eppinger [2][p 12] states:

> "A product development process is the sequence of steps or activities which an enterprise employs to conceive, design, and commercialize a product."

The text **Engineering Design** by Dym and Little [3][p 7] states:

> "Engineering design is the organized, thoughtful development and testing of characteristics of new objects that have a particular configuration or perform some desired function(s) that meets our aims without violating any specified limitations."

A special issue of the **Journal of Engineering Education** focused on engineering education research was published in January 2005. In the paper *Engineering Design Thinking, Teaching, and Learning* [11][p 104], Dym and Agogino *et al.* define engineering design to help determine what makes teaching design so difficult:

> "Engineering design is a systematic, intelligent process in which designers generate, evaluate, and specify concepts for devices, systems, or processes whose form and function achieve clients' objectives or users' needs while satisfying a specified set of constraints."

While many definitions of engineering design are offered, one common general premise is that of a process resulting in an end deliverable, product, or artifact. These definitions imply that the process is iterative, and that the end deliverable satisfies a given customer need. For the purposes of this thesis, the Dym and Agogino *et al.* definition [11] will be used. It implies that the process used is a thoughtful process, and that the driving force is to meet the needs of the customer, subject to constraints.

2.2 Engineering Design Processes

The processes used in engineering design have been documented in various forms since at least 1962 when Morris Asimow introduced a process in **An Introduction to Design** [1] (Figure 2.1). Processes have evolved through the years, as current embodiments are much simpler than the Asimow embodiment. These embodiments of the process have similarities between them, with the differences largely being in the level of specificity and terminology. In spite of these similarities, no one process has been universally adapted.

Several processes are examined and compared below, with the intent of highlighting the similarities and differences between them. The goal of this comparison is to provide a basis for the decision of which tools, or commonly occurring subprocesses, to include in the current study of information flow during the early stages of the overall design process. Ultimately the set of tools provided should be widely helpful, regardless of the process selected by any particular user or corporation.

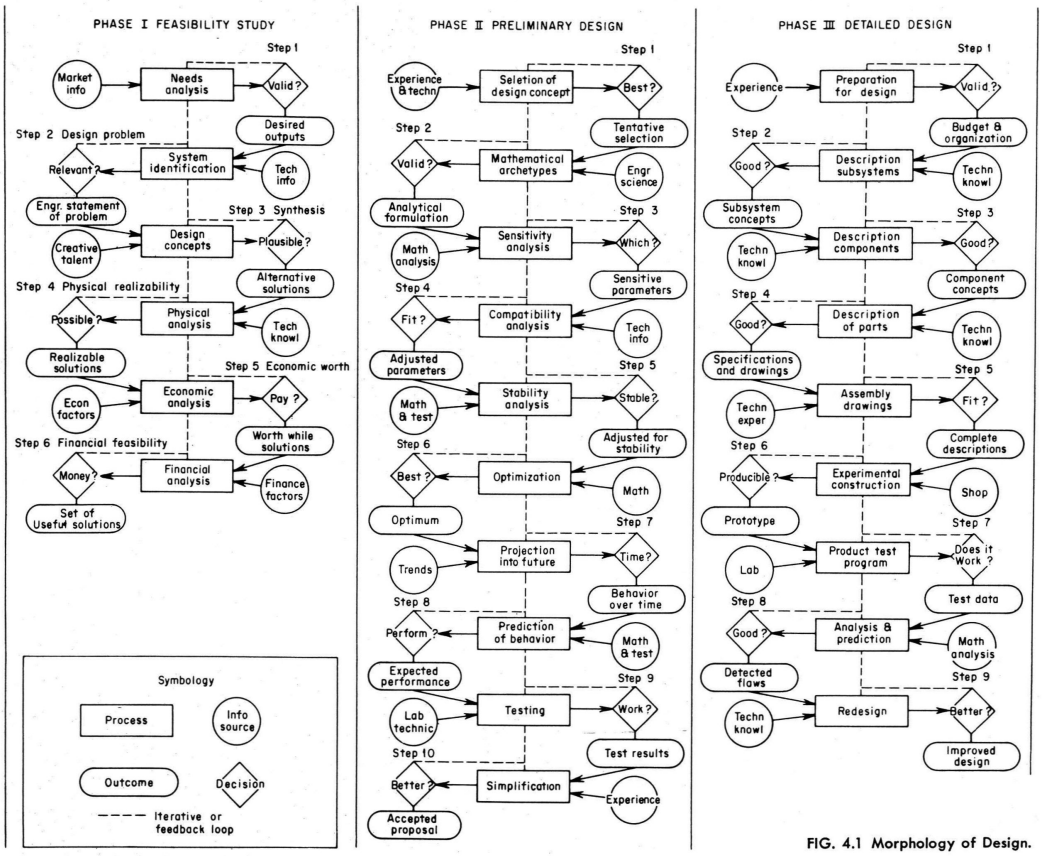


FIG. 4.1 Morphology of Design.

Figure 2.1: The Design Process Outlined in **An Introduction to Design** by Asimow [1][p 19]

2.2.1 Textbook Definitions

A variety of textbooks deal with defining and navigating the engineering design process. Each has its own terminology and tool set, and each has its own strengths. The four textbooks reviewed here were chosen for their use in the RIT curriculum and the variety of process models.

Ulrich and Eppinger

Product Design and Development by Ulrich and Eppinger [2] is the text currently being used for the Design Project Management (0304-730) and Multi-disciplinary Senior Design courses (0304-630, 0304-631) in the Kate Gleason College of Engineering (KGCOE) at Rochester Institute of Technology (RIT). Ulrich and Eppinger describe a six step generic product development process, shown in Figure 2.2.

- Planning

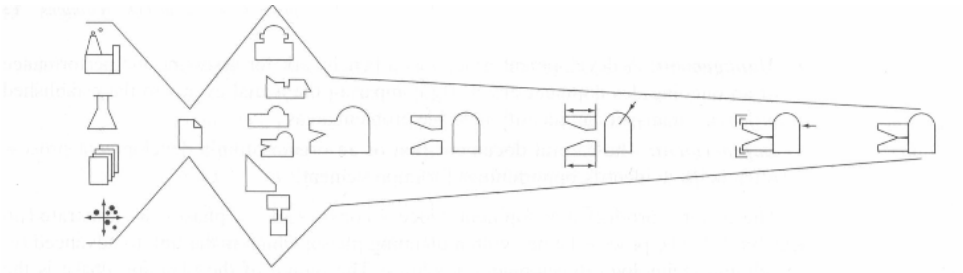
  The planning is prior to the launch of the actual product development process. This phase includes market objectives and technological development assessment. The five steps in product planning as stated in **Product Design and Development**[2][p 50] are:

  1. Identify opportunities.
  2. Evaluate and prioritize projects.
  3. Allocate resources and plan timing.
  4. Complete pre-project planning.
  5. Reflect on the results and the process.

  The deliverables of the planning phase are the identification of the target market for the product, business goals, key assumptions, and constraints.

- Concept Development

  "In the concept development phase, the needs of the target market are identified,



Phase 0: Planning	Phase 1: Concept Development	Phase 2: System-Level Design	Phase 3: Detail Design	Phase 4: Testing and Refinement	Phase 5: Production Ramp-Up
Marketing					
• Articulate market opportunity. • Define market segments.	• Collect customer needs. • Identify lead users. • Identify competitive products.	• Develop plan for product options and extended product family. • Set target sales price point(s).	• Develop marketing plan.	• Develop promotion and launch materials. • Facilitate field testing.	• Place early production with key customers.
Design					
• Consider product platform and architecture. • Assess new technologies.	• Investigate feasibility of product concepts. • Develop industrial design concepts. • Build and test experimental prototypes.	• Generate alternative product architectures. • Define major subsystems and interfaces. • Refine industrial design.	• Define part geometry. • Choose materials. • Assign tolerances. • Complete industrial design control documentation.	• Reliability testing. • Life testing. • Performance testing. • Obtain regulatory approvals. • Implement design changes.	• Evaluate early production output.
Manufacturing					
• Identify production constraints. • Set supply chain strategy.	• Estimate manufacturing cost. • Assess production feasibility.	• Identify suppliers for key components. • Perform make-buy analysis. • Define final assembly scheme. • Set target costs.	• Define piece-part production processes. • Design tooling. • Define quality assurance processes. • Begin procurement of long-lead tooling.	• Facilitate supplier ramp-up. • Refine fabrication and assembly processes. • Train work force. • Refine quality assurance processes.	• Begin operation of entire production system.
Other Functions					
• Research: Demonstrate available technologies. • Finance: Provide planning goals. • General Management: Allocate project resources.	• Finance: Facilitate economic analysis. • Legal: Investigate patent issues.	• Finance: Facilitate make-buy analysis. • Service: Identify service issues.		• Sales: Develop sales plan.	

Figure 2.2: Process Outlined in **Product Design and Development** by Ulrich and Eppinger [2][p 14]

alternative product concepts are generated and evaluated, and one or more concepts are selected for further development and testing." [2][p 13]

Tools identified for concept development include interviews, focus groups, hierarchical lists, and importance of needs surveys for identifying the customer need, target specifications and competitive benchmarking for establishing product specifications, problem breakdown into subproblems, literature searches, gallery methods, TRIZ, concept classification trees, and concept combination tables for concept generation, selection matrices and concept scoring for concept selection and concept surveys for concept testing.

- System-level design

  "The system-level design phase includes the definition of the product architecture and the decomposition of the product into subsystems and components." [2][p 15]

  Deliverables include a geometric layout, a functional specification of each subsystem, and a preliminary process flow diagram for the final assembly process.

- Detail design

  "The detail design phase includes the complete specification of the geometry, materials, and tolerances of all of the unique parts in the product and the identification of all of the standard parts to be purchased from suppliers." [2][p 15]

  Tooling for each component is designed and the process plan is laid out. The output of the phase is complete documentation of the design.

- Testing and refinement

  The testing and refinement phase is the prototype phase. Early prototypes determine whether the product fits the customer needs. Later prototypes address the manufacturing process.

- Production ramp-up

  The production ramp-up phase uses the intended production systems. The work force

is trained on the production process. Any bugs that arise need to be addressed before the transition to commercial production.

Ulrich and Eppinger [2] describe the concept development phase as the time when market needs are identified, concepts are generated and evaluated, and concepts are selected for further development and testing. While not specifically making use of an objective tree, Ulrich and Eppinger recommend organizing customer needs into a hierarchical list and establishing their relative importance. The objective tree format does display the information in a different graphical format, but the inherent grouping structure still exists for the hierarchical list. The text also deals with translating the customer needs into the engineering specifications.

Dym and Little

The Dym and Little text, **Engineering Design** was used in the ME Cornerstone Design course (0304-261) starting in academic year 2006-07. The design process used by Dym and Little [3] is as follows

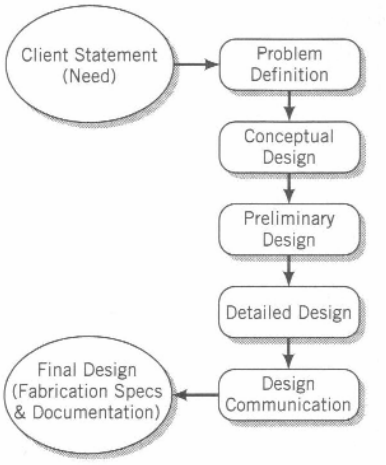


Figure 2.3: Process Outlined in **Engineering Design** by Dym and Little [3][p 23]

- Problem Definition

  Tasks under problem definition include clarifying design objectives, establishing user requirements, identifying constraints and establishing functions. Tools covered for completing these tasks include objectives trees, pairwise comparison charts, function-means trees, functional analysis, and requirements matrices. Means of gathering information include literature reviews, brainstorming, user surveys and interviews.

- Conceptual Design

  Tasks under conceptual design include establishing design specifications and generating design alternatives. Tools covered include performance specification, quality function deployment, and morphological charts. Means of gathering the necessary information include brainstorming, benchmarking, and reverse engineering.

- Preliminary Design

  Tasks in preliminary design include modeling, analyzing, testing, and evaluating conceptual designs. Tools include refined objective trees and pairwise comparison charts. Means of accomplishing the tasks include defining metrics, performing laboratory experiments, developing prototypes, running computer simulations and analysis, and conducting proof-of-concept testing.

- Detailed Design

  In detailed design, the chosen design is refined and optimized. Tools include discipline-specific CADD. Formal design reviews and beta testing are included in detailed design.

- Design Communication

  The completed design is documented in design communication. The deliverable is final fabrication specifications, along with the justification of the specifications.

This process focuses on the beginning stages of design and assumes that design is complete once the fabrication specifications have been defined and justified. Although the process does not specifically show it, the text does discuss using verification feedback between stages to repeat tasks if necessary, and using validation feedback to incorporate into the next design version. The text intentionally limits the detailed discussion to the earliest stages of design.

Ertas and Jones

The design process laid out in Ertas and Jones' textbook, **The Engineering Design Process** [4] is the design process (Figure 2.4) most similar to the FACETS process described in Chapter 3. While Dym and Little admittedly focus their text on the beginning stages of

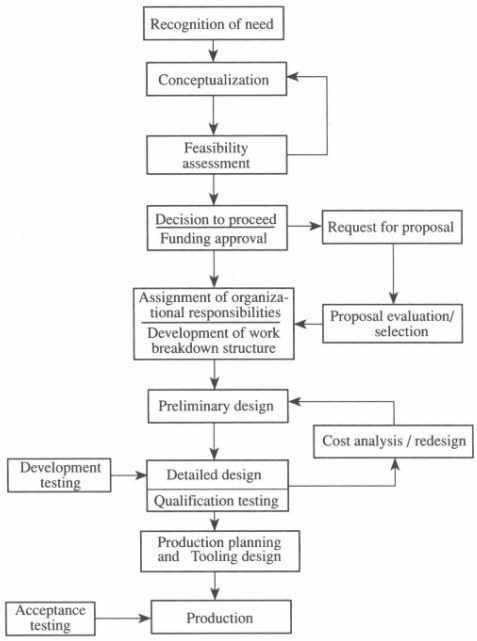


Figure 2.4: Process Outlined in **The Engineering Design Process** by Ertas and Jones [4][p 4]

design, the bulk of Ertas and Jones focuses on developing a selected concept. The design process described in Chapter 1 of **The Engineering Design Process** is not followed or used

after the authors present it.

Ford and Coulston

Ford and Coulston's text **Design for Electrical and Computer Engineers** is geared towards electrical and computer engineering students. The circular model (Figure 2.5) of the design process presented is different from the more linear models, with similar steps. The lines connecting the phases indicate that the process can flow between any phase, not

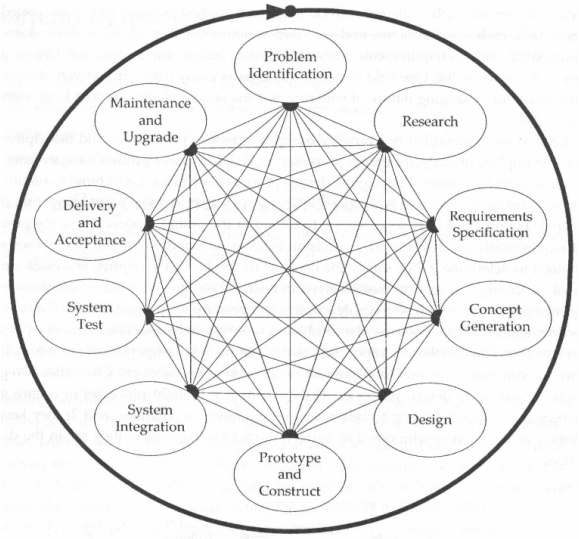


Figure 2.5: Process Outlined in **Design for Electrical and Computer Engineers** by Ford and Coulston [5][p 4]

just sequentially. Ford and Coulston state [p 4] "transitions between certain phases are unreasonable or very costly. It is virtually impossible to move directly from problem identification to system integration without developing a design concept first." As an aside to the text, if the lines between phases are interpreted to represent information flow rather than sequence of steps, then it may hold true that information can flow between any and all steps. For example, a novel feature in system integration could spark a new product idea, feeding information into a new problem identification.

Tools cited for needs identification follow the tools stated in Ulrich and Eppinger [2]; identify the customer needs, use an objective tree to hierarchically represent the needs, and create a relative ranking of the needs. Requirements specification tools include the House of Quality. Concept generation tools include a concept table, brainstorming, and nominal group technique as well as decision matrices for evaluating the concepts.

Summary

After a review of the design processes espoused in the texts above, a common theme seems to be the inclusion of a higher level of detail at the beginning of the design process. The tools described for accomplishing the early design phases have remarkable similarities, even if the names and formats are different.

2.2.2 Industry Processes

The "real world" does not always follow a text book. As students prepare to enter the work force, it is crucial to verify that the design processes being implemented in courses are relevant to what is currently being used in industry.

Two design practices commonly found in industry are a sequential style, and a concurrent style. Stage-Gate® [1], the typical sequential style, has set points or "gates" to clear before moving on to the next stage of work. In concurrent engineering, as much work is done in parallel as possible to reduce the overall time to market.

Stage-Gate®

A 1997 best practice study by the Product Development & Management Association revealed that almost 60% of leading U.S. product developers now use some type of Stage-Gate process [12]. The Stage-Gate process has several phases with a checkpoint or gate to clear before proceeding to the next phase (Figure 2.6).

[1] Stage-Gate® is a registered trademark of Product Development Institute Inc.

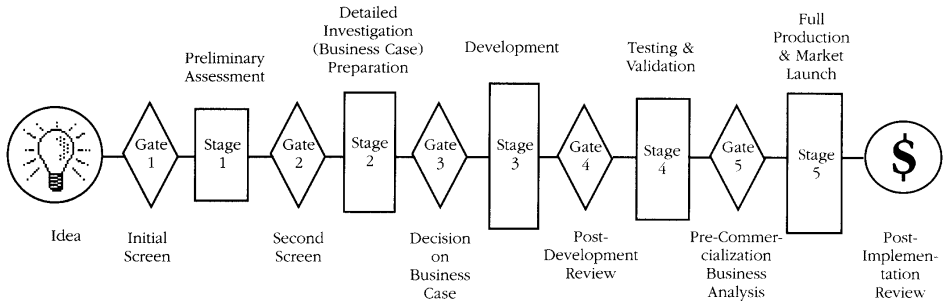


Figure 2.6: Stage-Gate Process [6][p 46]

- Gate 1 - Initial Screen
- Stage 1 - Preliminary Assessment

  A quick and inexpensive assessment of the technical merits of the project and its market prospects.

- Gate 2 - Second Screen
- Stage 2 - Detailed Investigation Preparation

  This is the critical homework stage - the one that makes or breaks the project. Technical marketing and business feasibility are accessed resulting in a business case which has three main components: product and project definition; project justification; and project plan.

- Gate 3 - Decision on Business Case
- Stage 3 - Development

  Business case plans are translated into concrete deliverables. The product development activities occur, the manufacturing or operations plan is mapped out, the marketing launch and operating plans are developed, and the test plans for the next stage are defined.

- Gate 4 - Post Development Review
- Stage 4 - Testing and Validation

  The purpose of this stage is to provide validation of the entire project: the product itself, the production process, customer acceptance, and the economics of the project.

- Gate 5 - Pre-Commercialization Business Analysis
- Stage 5 - Full Production and Market Launch

  Full commercialization of the product - the beginning of full production and commercial launch.

The paper *"A comparative study of six stage-gate approaches to product development"* by Phillips, Neailey and Broughton [7] shows how six different companies with six different product development processes can all be represented by a Stage-Gate process. The companies compared are Bombardier Aerospace Group, Kodak, General Electric, Lucas Industries, Rolls-Royce, and Motorola. These companies utilize between four and ten phases, which the Phillips *et al.* argue can be grouped into the four generic stages as shown in Figure 2.7.

One of the differences between Stage-Gate and typical textbook processes is that Stage-Gate is more targeted at the business economics. More focus is given in Stage-Gate to justifying the business case, since a concept that doesn't satisfy the customer need is most likely a bad business decision.

Concurrent Engineering

Concurrent Engineering (CE) is a management/operational approach which aims to improve product design, production, operation, and maintenance by developing environments in which personnel from all disciplines (design, marketing, production engineering, process planning, and support) work together and share data throughout all phases of the product

	Organisation					
Generic stages	**Bombardier**	**Kodak**	**General Electric**	**Lucas Industries**	**Rolls-Royce**	**Motorola**
Preliminary concept development	Conceptual design Management review and approval	Customer mission Technical demonstration	Customer needs Concept Feasibility	Opportunity evaluation	Project planning	Product definition
Design and development	Preliminary definition	Technical/ operational feasibility	Preliminary design Critical productivity	Design and development	Full concept definition	Contract development
Validation	Detail definition Product definition release Product certification	Capability demonstration Product/ process design Acceptance and production	Market/field test Manufacturing feasibility Market readiness	Validation Implementation Manufacturing	Propulsion system realisation	Development through manufacturing start-up
In-service product support	Program completion	No equivalent	Market introduction follow up	Support	In-service monitoring and technical support	Program wrap-up
Number of phases	7	6	10	5	4	4
High or low phased	High	High	High	Low	Low	Low

Figure 2.7: Stage Gate Process Comparison [7][p 294]

life cycle. This is in contrast to the traditional sequential style of product development, where design tasks follow in sequence. A comparison between the two design procedures is shown in Figure 2.8.

Toyota is one of the most benchmarked companies that uses a concurrent engineering process. Some of their design practices tend to go against typical design methodologies. Instead of the voice of the customer, the main voice is that of the chief engineer who has been groomed to make critical decisions. Written communication is preferred to meetings, which are held only when written communication is inadequate. Engineers belong to functional groups and are loaned to project teams when needed. Instead of a structured process, Toyota uses a simple timeline with strict deadlines. The book **Concurrent Engineering Effectiveness** by Fleischer and Liker [8][p 47] establishes that a work process is critical to supporting the core work.

> The "work processes" of product development are the set of activities and the connection between those activities that are used to develop a product and the processes for making it. They include:
>
> - activities

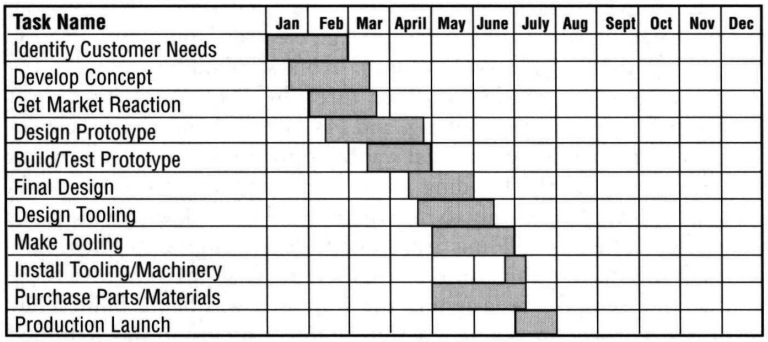


Figure 1.3 • Overlapping or Concurrent Product Development Process

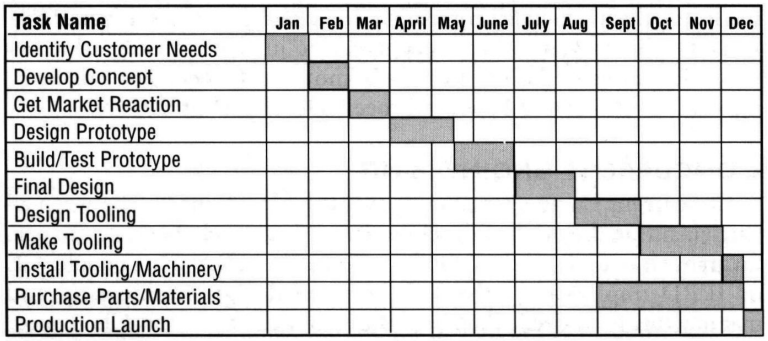


Figure 1.4 • Sequential or Waterfall Product Development Process

Figure 2.8: Comparison Between Concurrent Engineering (top) and Sequential Engineering (bottom) Design Processes. [8][p 8]

- flow of information and physical objects between activities
- ordering and timing of activities
- control mechanisms

Fleischer and Liker suggest that these elements should exist in any work process model and an appropriate model should be chosen based on individual company needs.

Activities are "the actual work that adds value to the product, *e.g.*, creating specifications, collecting information on customer wants, making a sketch of a mechanical part, creating a 3-D model, building a prototype." [8][p 47] Flow of information between activities includes written and verbal instructions, detailed drawings, computer models, physical mock-ups or prototypes, clarification questions, quotes, purchase orders and confirmations. Some activities need information or resources before they can take place and other activities can take place in parallel. Addressing timing in the work process model keeps the time line in check. Control mechanisms are "used to ensure that activities remain aligned and that the project remains on target." [8][p 49] Reporting and reviewing activities help make sure that activities and information flow happen. A model of the work process needs to include all four of these elements to be considered complete.

The work process used for any design team should be based on three factors[p 69].

1. *How detailed should the model be?* Should the model include every piece of information associated with the model, or should it be more of an outline in which everyone "knows" what to do?
2. *Who does the work of modeling?* The decision comes down to whether the task is done from the top, with a high level view dominating, or whether it should be done in a distributed manner with each level creating its own model.
3. *How standard should the model be?* Should the model change for each project, *i.e.*, be customized, or should the model attempt to stay as-is for every project?

Fleischer and Liker recommend a model that is detailed enough for everyone to understand it and be able to follow it. Extremely detailed sets of procedures can be overkill.

The work of developing the model itself should be distributed. A simpler model for the entire company can be developed by top level managers, but more detailed models should be developed by the people doing the work. With the people who built the models also implementing them, there is a greater likelihood that the model will be useful and used. One inflexible model is not recommended, as projects are unique, yet creating a new process for every individual project is a waste of time. A customizable process allows for variation between projects, and also variation over time based on lessons learned. The minimum recommendation by Fleischer and Liker is to use a high-level phases and gates process combined with a schedule, with more detailed phases, gates and schedules at lower level to meet the requirements of the level above. When developing an appropriate model, keep in mind the four stated product development process principles[p 324];

1. A structured design process should be developed and used.
2. The voice of the customer should be the focus throughout the process.
3. Stable strategic targets should be developed early in any design project.
4. Design history should be captured and re-used. Reinventing the wheel is obviously wasteful.

As stated in the processes evaluated here and many more [13, 14, 15], the early facets of many design processes can be summarized as encompassing the identification of the customer needs and developing product concepts that will satisfy those customer needs. These facets will be referred to herein as “Needs Assessment” and “Concept Development”. Every process reviewed herein includes these facets, and they are always included early in the design process. In describing these facets, several useful tools were mentioned, which are reviewed in Section 2.3.

2.3 Design Tools

Within the early stages of design, several tools are identified for helping to capture the customer need, identify the functions of the design, develop a design space of means, and narrow that design space. Before tackling a design, it is necessary to understand the reasons driving the design. Taking time at the beginning of the design process to verify that the correct issue is being addressed and understanding what will provide value to the customer can make a vast difference in the results.

Several methods can, and should, be employed to understand the customer needs. A variety of tools are commonly used to capture the needs of the customer, organize the needs, and translate them into specifications, identify design functions, and develop concepts. These tools include, but are not limited to:

- Affinity Diagram
- Brain Ball
- Brainstorming
- C-Sketch Method
- Function-Means Tree
- House of Quality
- Morphological Chart
- Objective Tree Method
- Pairwise Comparison
- Weighted Voting

The tools described here are a basis for the scripts developed for this thesis project. Those scripts will be discuss in Section 4.3.

2.3.1 Affinity Diagram

Affinity diagrams, also known as the KJ method after it's creator Jiro Kawakita, are a way of organizing a collected set of data [16]. Affinity diagrams are typically used if there are a large number of ideas, complex issues, or if it's necessary to reach agreement in a group setting. A procedure for preparing an affinity diagram is presented in **Quality Toolbox** [16][p 96]:

1. Record each item with marking pens on separate sticky notes. Randomly spread notes on a large work surface so all notes are visible to everyone. The entire team gathers around the notes and participates in the next steps.
2. *It is very important that no one talk during this step.* Look for ideas that seem to be related in some way. Place them side by side. Repeat until all notes are grouped. It's okay to have "loners" that don't seem to fit a group. It's alright to move a note someone else has already moved. If a note seems to belong in two groups, make a second note.
3. *You can talk now*. Participants can discuss the shape of the chart, any surprising patterns, and especially reasons for moving controversial notes. A few more changes may be made. When ideas are grouped, select a heading for each group. Look for a note in each grouping that captures the meaning of the group. Place it at the top of the group. If there is no such note, write one. Often it is useful to write or highlight this note in a different color.
4. Combine groups into supergroups if appropriate.

Approximately twenty students in Design Project Management (DPM) at RIT were required to individually attend a variety of thirty minute design project technical reviews being presented by students enrolled in Multidisciplinary Senior Design. The DPM students were subsequently asked to create an affinity diagram based on best practices observed in the technical reviews. The DPM students were given ten minutes to individually document their thoughts on the technical reviews on sticky notes. After each student finished collecting their thoughts, the notes were randomly placed on a large white pad (Figure 2.9). A

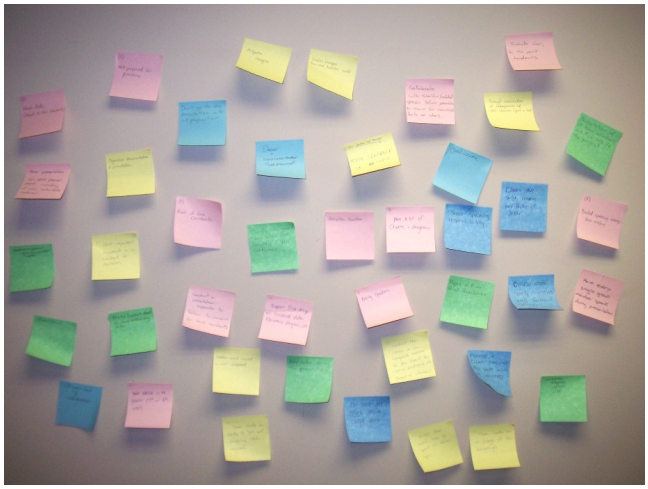


Figure 2.9: Unsorted Affinity Diagram From Design Project Management

small group of students then silently worked to group the notes. After agreeing on the correct groups, the team identified headings for each of the groups (Figure 2.10). The headings that came out for best practices in senior design technical reviews and some representative comments under the headings were:

Visual Aids / Power Point Prepare a power point presentation. Good volume. Have a lot of charts and diagrams.

Shared Speaking Roles Share speaking responsibility. Dressed in a business manner. Presentation of all team members and their role in the project.

Preparation for Questions Don't go to the presentation without preparation. Organized & well rehearsed. Well prepared for questions.

Leaders Role Leader well versed in each component. Team leader be ready to "bail out" struggling team members.

Presentation Agenda Provide a clear timeline for what will be discussed. Know how much time to spend on each topic.

Figure 2.10: Sorted Affinity Diagram From Design Project Management

Documentation for Audience Printed handout / report for viewer to follow along with. Include clear, to the point, handouts.

There were also a few ungrouped items such as “clearly state safety concerns”, “factor of safety”, and “thorough examination of consequences of each decision”.

The benefit of using an affinity diagram is in moving outside of any preconceived categories of ideas and bringing a large group of individuals to consensus. To be successful, the step of grouping the notes needs to be given plenty of time to allow themes to surface.

2.3.2 Brainball

Brainball is a group activity for quickly generating a large number of spontaneous ideas. The general purpose is to keep the thoughts coming quickly. Therefore some of the ideas aren’t exactly in line with the focus. That’s fine. Someone in the group needs to act as the recorder of ideas, as they will come quickly. To perform a brainball session, the steps are:

1. The team leader reviews the focus of the brainball session.

2. A soft object, such as a foam ball, is thrown to a random team member. The first team member thinks of the first idea in their head that starts with the letter A.

3. After the first team member speaks, they toss the ball to another team member to come up with an idea that starts with the letter B.

4. This continues through the entire alphabet.

5. Review the session and document any ideas that are discussed.

The hope is that something that is said in the Brainball session may spark some thought in one of the team members.

A brainball session was held in the Cornerstone Design course in spring of 2006-2007 academic year. The focus area was designing a water heating system for a new home. The results of their brain ball session are shown in Table 2.1. While some of the responses are

Table 2.1: Brainball Session Results

A] aluminum	N] nuclear
B] big tank	O] osmosis
C] cats	P] platinum
D] dark	Q] Queen of England
E] electric fence	R] regeneration
F] ferromagnetic coil	S] super conductive coil
G] gas	T] tropical rainforest
H] hydrolyser	U] ultraviolet rays
I] injection	V] very expensive ball
J] jello	W] water heating
K] kangaroos	X] x-ray
L] lake	Y] youth
M] mass	Z] zeta potential

obviously not productive, such as the “Queen of England”, others such as “big tank” and “water heating” were ultimately considered in the final proposed design.

2.3.3 Brainstorming

> "It is easier to tone down a wild idea than to think up a new one." - Alex Osborn

The term "brainstorm" was coined by Alex Osborn in 1939, and was first published in his book, **Your Creative Power** [17] in 1948. This technique for harnessing the creative power of a group of people recognized that simply gathering people to generate ideas was not productive. The tendency was to judge existing ideas rather than create and add ideas. To avoid these negative tendencies, Osborn laid out four simple ground rules for group brainstorming.

1. Judicial judgement is ruled out. Criticism of ideas must be withheld until the next day.
2. "Wildness" is welcomed. The crazier the idea the better; it's easier to tone down than to think up.
3. Quantity is wanted. The more ideas we pile up, the more likelihood of winners.
4. Combination and improvement are sought. In addition to contributing ideas of our own, let's suggest how another's idea can be turned into a better idea; or how two or more ideas can be joined into still another idea.

Other tips for successful brainstorming are to assemble five to ten people, invite members that represent diverse experiences and include at least a few self starters. It's also very important to state a specific topic for the brainstorming session. With too broad of a topic, it's too easy to lose momentum.

Recent research in brainstorming indicates that group dynamics can interfere with the effectiveness of brainstorming. While rules try to minimize the criticism of crazy ideas, the fear of making out of the box suggestions among coworkers can reduce the number of suggestions made. Paulus, *et al.* [18] recommends adding four new rules into Osborn's

original four rules. These rules were published by Oxley *et al.* [19] to minimize unproductive chatter, and keep the session moving. The additional rules, with the numbering adjusted here, are

> (5) Do not tell stories or explain ideas; (6) when no one is saying ideas, restate the problem and encourage one another to generate more ideas; (7) encourage those who are not talking to make a contribution; (8) suggest that participants reconsider previous categories when they are not generating many more new ideas.

The book **Groups: Interaction and Performance** [20] has been cited for suggesting that groups do not have the creative power that an individual has. This isn't to say that group brainstorming activities are useless, but more guidance and structure should be laid out ahead of time to best make use of the group brainstorming session. Improvements to the typical brainstorming session include ways to protect the feelings of the participants. This psychological safety will help the participant feel safe enough to open up and offer the more creative, although not necessarily "accepted" ideas. These are the ideas that drive the creative cycle and help encourage more ideas.

2.3.4 C-Sketch Method

Collaborative sketching (C-sketch) is a graphical method for sharing concept ideas within a group [3]. This activity is completed in silence.

1. Six (recommended) people sit around a table. Review and clarify the focus of the ideation session.
2. Hand each group member a blank piece of paper.
3. Each group member sketches their idea.
4. After five minutes has elapsed, each team member passes their drawing to the team member on the right. No speaking is allowed to clarify what the previous member has sketched.

5. The next team member then expands on the already sketched design. Further detail should be added as the drawings circulate around the group.

6. After the drawing gets back to the original sketcher, the group should share and review all of the sketches.

The benefit of using the C-sketch method is that everyone's ideas are represented. It's also helpful in that it forces each team member to contribute to methods that they didn't champion.

Yang and Cham [21] examine the role of sketching in the early stages of design. Among the conclusions drawn is that the quality of sketches does not indicate a level of mechanical recall or overall performance. As participants engage in collaborative sketching, they should not be nervous or embarrassed by their level of sketching ability.

2.3.5 Function-Means Tree

A function-means tree helps determine the basic and secondary functions of a design, along with possible ways of accomplishing those functions [3]. The top level is the very basic function to be met. For the example shown in Figure 2.11, notice that the basic function isn't cigarette lighter, or even light cigarettes. The function, "ignite leafy materials" is as basic and non-specific as possible. After identifying the basic function, there are several methods or means that can be used for satisfying that function. Depending on which means is used, the secondary functions will vary. Some functions will show up under all or many of the means, and are likely inherent to the problem of the design. Other functions will only need to be addressed if a certain means is chosen.

To create a useful function-means tree, it's critical to not simply populate the tree to justify a preconceived solution. It should also not be the only tool used in determining a design space.

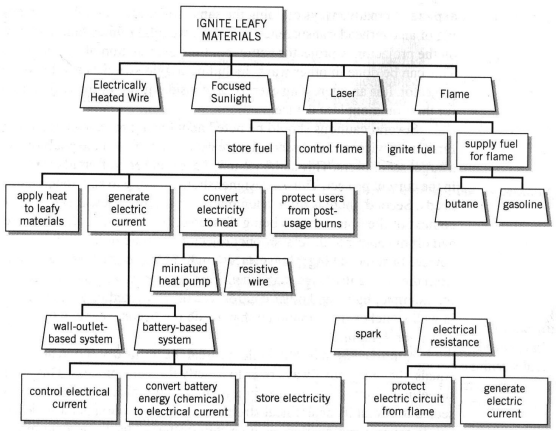


Figure 2.11: Function-Means Tree From **Engineering Design** by Dym and Little [3][p 85]

2.3.6 House of Quality

The House of Quality (HoQ) is the diagrammatic result of performing quality function deployment (QFD) [16]. The QFD method organizes the necessary pieces of information for understanding the problem. **The Mechanical Design Process** [9] walks through step by step taking a HoQ skeleton (Figure 2.12) and filling it in (Figure 2.13).

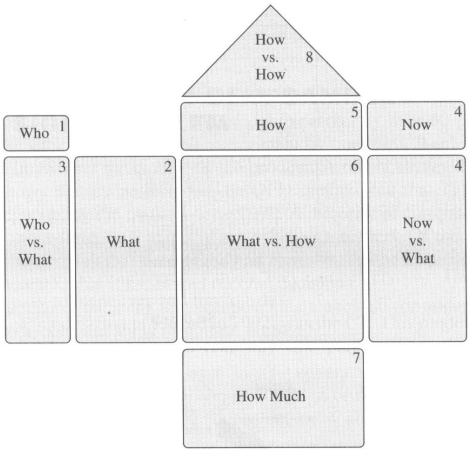
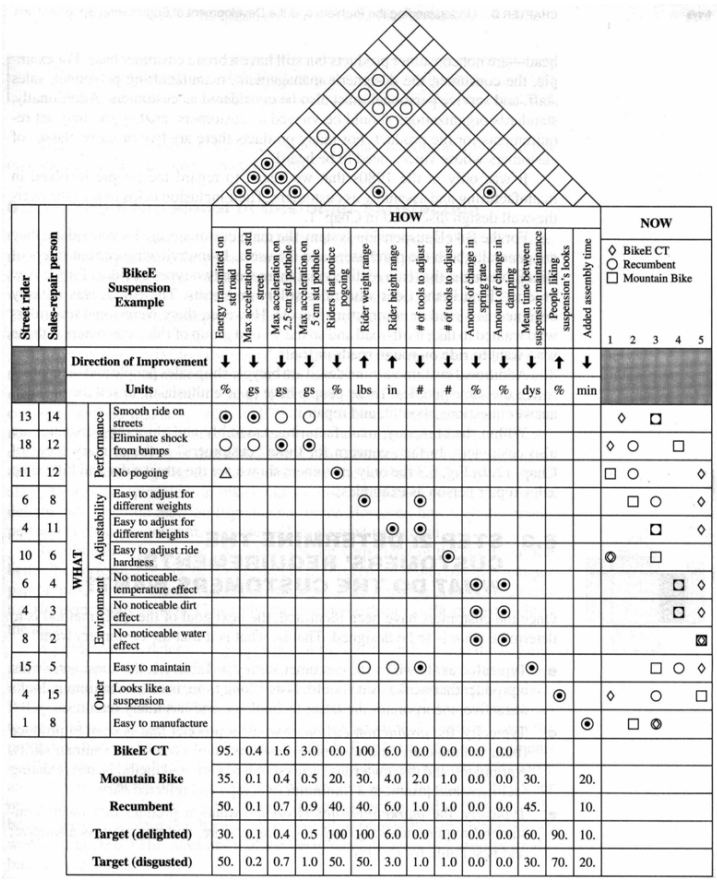


Figure 2.12: House of Quality Structure From **The Mechanical Design Process** by Ullman [9][p 116]



Street rider	Sales-repair person			BikeE Suspension Example	Energy transmitted on std road	Max acceleration on std street	Max acceleration on 2.5 cm std pothole	Max acceleration on 5 cm std pothole	Riders that notice pogoing	Rider weight range	Rider height range	# of tools to adjust	# of tools to adjust	Amount of change in spring rate	Amount of change in damping	Mean time between suspension maintainance	People liking suspension's looks	Added assembly time	NOW: ◇ BikeE CT ○ Recumbent □ Mountain Bike (1 2 3 4 5)
				Direction of Improvement	↓	↓	↓	↓	↓	↑	↑	↓	↓	↓	↓	↓	↑	↓	
				Units	%	gs	gs	gs	%	lbs	in	#	#	%	%	dys	%	min	
13	14	WHAT	Performance	Smooth ride on streets	⊙	⊙	○	○											◇ □
18	12			Eliminate shock from bumps	○	○	⊙	⊙											◇ ○ □
11	12			No pogoing	△				⊙										□ ○ ◇
6	8		Adjustability	Easy to adjust for different weights						⊙		⊙							□ ○ ◇
4	11			Easy to adjust for different heights							⊙	⊙							□ ◇
10	6			Easy to adjust ride hardness									⊙						◎ □
6	4		Environment	No noticeable temperature effect										⊙	⊙				□ ◇
4	3			No noticeable dirt effect										⊙	⊙				□ ◇
8	2			No noticeable water effect										⊙	⊙				◎
15	5		Other	Easy to maintain						○	○	⊙				⊙			□ ○ ◇
4	15			Looks like a suspension													⊙		◇ ○ □
1	8			Easy to manufacture														⊙	□ ◎
				BikeE CT	95.	0.4	1.6	3.0	0.0	100	6.0	0.0	0.0	0.0	0.0				
				Mountain Bike	35.	0.1	0.4	0.5	20.	30.	4.0	2.0	1.0	0.0	0.0	30.		20.	
				Recumbent	50.	0.1	0.7	0.9	40.	40.	6.0	1.0	1.0	0.0	0.0	45.		10.	
				Target (delighted)	30.	0.1	0.4	0.5	100	100	6.0	0.0	1.0	0.0	0.0	60.	90.	10.	
				Target (disgusted)	50.	0.2	0.7	1.0	50.	50.	3.0	1.0	1.0	0.0	0.0	30.	70.	20.	

Figure 2.13: Completed House of Quality From **The Mechanical Design Process** by Ullman [9][p 117]

The step by step procedure for completing a House of Quality as described in **Quality Toolbox** [16][p 306] is:

1. Assemble a cross-functional team. The team should include people knowledgeable about the customer and others knowledgeable about the product or service. In each of the following steps, the team must first obtain the information needed to build the house. This information may come from direct contacts with customers, from departmental studies (for example, marketing, engineering), or from the team members' job knowledge.

2. Write customer requirements in the left wing of the house as row labels. As much as possible, the customers' own wording should be used.

3. Add a column between the row labels and the center of the house for importance. 1 to 10 is a commonly used numerical scale, where 1 is unimportant and 10 is extremely important. Assign a number to each customer requirement, based on information you have obtained from customers themselves.

4. In the right wing of the house, record information about customers' perception of existing comparable products or services, your own and competitors'. Usually a scale of 1 to 5 is used. Use different symbols for your product and for each of the competitors' products.

5. (Optional) You may choose to add additional information about customer requirements. Information sometimes included: customer complaints, sales points, target value for competitive position, scale-up factor, absolute weight.

6. In the attic of the house, write product or service characteristics as a column headers. Choose characteristics that directly affect the customer requirements and are measurable. Word them in the technical terms used within your organization.

7. Add another row in the attic with symbols indicating whether the characteristic needs to increase or decrease to better meet customer requirements. Common symbols are $+$ and $-$ or $\uparrow$ and $\downarrow$.

8. (Optional) You may choose to add more information about product or service characteristics. Information sometimes included: cost, cost of servicing complaints, technical difficulty.

9. In the center of the house, use the matrix between customer requirements (rows) and product or service characteristics (columns) to identify relationships between them. Use symbols to indicate whether a relationship between the requirement and the characteristic is positive or negative and how strong it is.

10. On the roof of the house, use the matrix to identify correlations between product or service characteristics. Use symbols indicating whether a relationship is positive or negative and how strong it is.

11. In the basement of the house, create a row identifying the measurement unit for each of the product or service characteristics.

12. Also in the basement of the house, record data on the performance of existing comparable products or services, your own and competitors'. Or you can use a relative scale of 1 to 5 and different symbols for each organization, as the customers' assessments were done.

13. (Optional) Determine weights for each of the product or service characteristics. Assign a numerical scale to the relationship symbols in the center of the house. Commonly used scales for weak, strong and, very strong relationships are 1, 3, 5 or 1, 3, 9. Starting with the first column, multiply each relationship number by the importance (or absolute weight, if used) of that customer requirement. Add the results down the entire column. The sum is the weight for that characteristic. In the next row, rank-order the product or service characteristics, starting with 1 for the characteristic with the highest weight.

14. Determine targets for the measurements of each product or service characteristic. Synthesize all the information in the house of quality to decide on appropriate targets for the new design. Record the targets in a row in the basement of the house.

The recent publication, *"Enhancing the Quality Function Deployment Conceptual Design Tool"* [22], reaffirms the positive influence of QFD on the quality of engineering design outcomes, but addresses the aspects of QFD that make it initially difficult for an engineering student to grasp. Modifications suggested include clarifying confusing nomenclature, segmenting the house to visibly separate information, and replacing the current strength of relationship symbols to be able to indicate if the relationship is a positive or negative relationship.

2.3.7 Morphological Chart

Morphological charts are a way to mix and match means to accomplish functions in order to investigate possible design combinations [3]. The morphological chart doesn't require any additional input or information. The procedure for creating a morphological chart may be summarized as

1. List the required functions of the design down the left hand side of the chart.
2. List all currently proposed means for satisfying that function to the right. If the means have been ranked at all, then the means should be listed in rank order.

By displaying all of the means and functions on one chart, it becomes recognizable that there are a large number of design combination possibilities. For the example shown from **Engineering Design** [3] (Figure 2.14), the number of design combinations is $4 \times 6 \times 5 \times 3 \times 2 = 720$. Not all of these design possibilities are feasible. For example, "tear corner" as a means of providing access wouldn't work in combination with a "glass bottle". Not all of the design combinations need to be explored, but the morph chart is a simple tool for generating more combinations than are originally generated through the stand-alone tools such as brainstorming. While brainstorming is a tool for generating means to accomplish a single function, the morph chart is a tool for expanding the design space.

2.3.8 Objective Tree

An objective tree provides a graphical representation of the needs or objectives of the design project. The top of the tree would be the primary objective of the part (*ie*; Safe Beverage Container). The second level expands on the general category of objectives (safe, promote sales). The third layer expands the objectives. The levels continue as needed until all the needs of the customer are listed. While it is helpful to have the needs already identified, it's quite plausible to see how filling the objective tree will help to identify needs not previously listed. The objective tree procedure summarized by Cross [23][p 54] is

MEANS FEATURE/FUNCTION	1	2	3	4	5	6
Contain beverage	can	bottle	bag	box	••••	••••
Material for drink container	aluminum	plastic	glass	waxed cardboard	lined cardboard	mylar films
Mechanism to provide access to juice	pull tab	inserted straw	twist top	tear corner	unfold container	••••
Display of product information	shape of container	labels	color of material	••••	••••	••••
Sequence manufacture of juice container	concurrent	serial	••••	••••	••••	••••

Figure 2.14: Sample Morphological Chart From **Engineering Design** by Dym and Little [3][p 106]

1. Prepare a list of design objectives. These are taken from the design brief, from questions to the client, and from discussion in the design team.

2. Order the list into sets of higher-level and lower-level objectives. The expanded list of objectives and sub-objectives is grouped roughly into hierarchical levels.

3. Draw a diagrammatic tree of objectives, showing hierarchical relationships and interconnections. The branches (or roots) in the tree represent relationships which suggest means of achieving objectives.

An example from Dym and Little [3] is the "Safe Beverage Container" which is covered in detail in the case study chapter. Following the three step procedure outlined, first the design objectives are listed (Figure 2.15). The second step of grouping the attributes and the third step of drawing the tree are both shown in Figure 2.16.

Looking at a completed objective tree, approaching it from the top down, lower levels should address how upper levels are achieved. Looking at it from the bottom up, the upper levels should address why the lower levels are necessary. The objective "Easy for kids to use" is a means by which the product satisfies needs to be easy to open and hard to spill. "Easy for kids to use" is also one way of achieving appeals to parents. Objectives that don't fit into the tree may not address the needs of the customer, or a need of the customer may

Safe	→	DIRECTLY IMPORTANT
Perceived as Safe	→	Appeals to Parents
Inexpensive to Produce	→	Permits Marketing Flexibility
Permits Marketing Flexibility	→	Promotes Sales
Chemically Inert	→	*Constraint* on Safe
Distinctive Appearance	→	Generates Brand Identity
Environmentally Benign	→	Safe
Environmentally Benign	→	Appeals to Parents
Preserves Taste	→	Promotes Sales
Easy for Kids to Use	→	Appeals to Parents
Resists Range of Temperatures	→	Durable for Shipment
Resists Forces and Shocks	→	Durable for Shipment
Easy to Distribute	→	Promotes Sales
Durable for Shipment	→	Easy to Distribute
Easy to Open	→	Easy for Kids to Use
Hard to Spill	→	Easy for Kids to Use
Appeals to Parents	→	*Promotes Sales*
Chemically Inert	→	*Constraint* on Preserves Taste
No Sharp Edges	→	*Constraint* on Safe
Generates Brand Identity	→	Promotes Sales
Promote Sales	→	DIRECTLY IMPORTANT

Figure 2.15: Attribute List for the Safe Beverage Container From **Engineering Design** by Dym and Little [3][p 57]

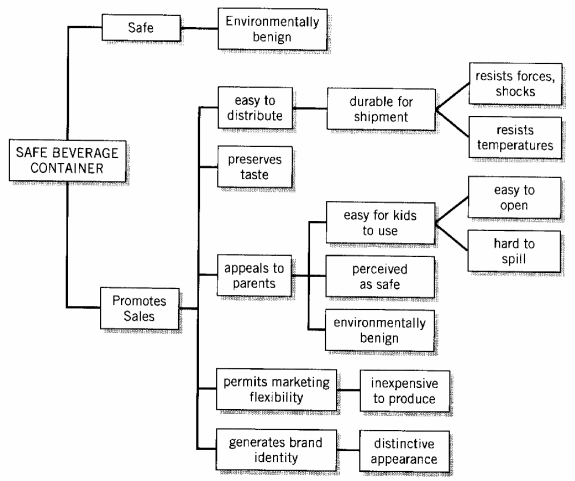


Figure 2.16: Objecive Tree for the Safe Beverage Container From **Engineering Design** by Dym and Little [3][p 58]

be going unmet.

2.3.9 Pairwise Comparison

Pairwise comparisons help determine the relative importance between two items being compared. This is completed by providing the user with two options and asking the user to select which option is of greater importance, or indicate that they are equally important. The information is displayed in a chart, and a tally is maintained to display the importance of each item.

Here is one pairwise comparison procedure from **Quality Toolbox** [16][p 373]

1. "List options as row labels down the left side of a an L shaped matrix. Number or letter each option sequentially. For column headings, write all pairs of comparisons. To do this methodically, pair each option with all succeeding options. If there are n options, there will be $n \times (n-1)/2$ pairs.
2. Start with the first column, option A versus option B. Each group member votes for one of these two options. Votes are tallied in the two rows opposite these two options.
3. Continue to vote one column at a time until all pairs have been voted upon.
4. Add across each row to obtain the total votes for each option. (Each column's sum should equal the number of people voting.) If the top vote-getters are close, check to be sure the comparison of those options matches the results of the total vote."

Dym, Wood, and Scott [24] argue for the validity of pairwise comparison charts which were called into question with respect to the Arrow Impossibility Theorem, which states that it is impossible to select a "fair" design or attribute if there are more than two to choose from [3]. They state [24][p 2]

"A voting procedure can be characterized as fair if five axioms are obeyed:

1. *Unrestricted*: All conceivable rankings registered by individual voters are actually possible.

2. *No Imposed Orders*: There is no pair A, B for which it is impossible for the group to select one over the other.
3. *No Dictator*: The system does not allow one voter to impose his/her ranking as the group's aggregate ranking.
4. *Positive Response*: If a set of orders ranks A before B, and a second set of orders is identical to the first except that individuals who ranked B over A are allowed to switch, then A is still preferred to B in the second set of orders.
5. *Independence of Irrelevant Alternatives* (IIA): If the aggregate ranking would choose A over B when C is not considered, then it will not choose B over A when C is considered."

When these five conditions hold true, Dym, Wood, and Scott state that the results of the pairwise comparison can be assumed to be a fair result. See [25] and Osei-Bryson [26] also deal with the issues that arise with respect to Arrow's Theorem.

Dym and Little [3] recommend completing the chart in a constrained, top-down fashion. Doing so ensures that items being compared are at the same level in the a tree, and that the higher abstract items are compared before the details.

After the items being compared are selected, those items are written across the top of a chart, and along the left hand side of a chart. Each time an item is being compared to itself (along the diagonal) a dash is entered. To fill out the rest of the chart, work across each row, comparing that row's item with all of the column items. If the row is more important than the column, then a 1 is entered in the cell. If the column is more important, a 0 is entered. For equal importance, enter a $1/2$. Fill out the entire chart in this fashion. Once all cells are entered, making sure there aren't any conflicts (For example, the row for item A has a 1 vs. item B, but the row for item B has a 1 vs. item A), the values are summed across the rows. The scores create a rank ordering. The value of the score has no weight.

An example from **Engineering Design** [3] shows two charts (Figure 2.17) for the same design, but two different customers - signifying that not everyone will agree on the importance of design attributes. The objectives chosen to use in the pairwise comparison chart

Goals	Environ. Benign	Easy to Distribute	Preserve Taste	Appeals to Parents	Market Flexibility	Brand ID	Score
Environ. Benign	••••	0	0	0	0	0	0
Easy to Distribute	1	••••	1	1	1	0	4
Preserve Taste	1	0	••••	0	0	0	1
Appeals to Parents	1	0	1	••••	0	0	2
Market Flexibility	1	0	1	1	••••	0	3
Brand ID	1	1	1	1	1	••••	5

(a) GRAFT's weighted objectives

Goals	Environ. Benign	Easy to Distribute	Preserve Taste	Appeals to Parents	Market Flexibility	Brand ID	Score
Environ. Benign	••••	1	1	1	1	1	5
Easy to Distribute	0	••••	0	0	1	0	1
Preserve Taste	0	1	••••	1	1	1	4
Appeals to Parents	0	1	0	••••	1	1	3
Market Flexibility	0	0	0	0	••••	0	0
Brand ID	0	1	0	0	1	••••	2

(b) BJIC's weighted objectives

Figure 2.17: Pairwise Comparisons From **Engineering Design** by Dym and Little [3][p 65]

were based on their location in the objective tree (Figure 2.16).

2.3.10 Weighted Voting

Weighted voting is a tool used to rank items. It is commonly used to rank the design team members' preference for concept selection. It is preferred over a straight vote due to its ability to highlight concepts that are acceptable to all members, but may not always be the top choice.

Below is a summary of the procedure from **Quality Toolbox** [16][p 360]

1. Display the list of options, combining duplicate items.
2. Number (or letter) all items.

3. Decide also how many votes each member will have. Usually five votes are allowed. The longer the original list, the more votes will be allowed, up to 10.

4. Members distribute these points over their choices. For example, one person may assign all votes to one choice about which that person cares fervently, while another may distribute votes equally among top three choices.

5. Tally votes.

6. If a decision is clear, stop here. Otherwise, continue with a brief discussion of the vote.

7. Repeat the voting process of steps 4 and 5 if necessary.

It is important to remember that the weighted voting tool is purely subjective and dependent on the opinions of individual team members.

The tools documented here represent only a portion of the tools available for dealing with the early facets of design. The ones chosen to be discussed reflect the tools used in courses in the Mechanical Engineering design sequence at RIT. Textbooks referenced give much more in-depth instruction on each of the tools.

The thesis project will lay down the ground work associated with providing users with a web based interface to these tools. The web based interface will allow design teams to store data, as well as flow information between the given tools. For example, information collected through the brainstorming tools can be organized into an affinity diagram and / or ranked using a weighted voting tool. Also, the affinity diagram created could be rendered as an objective tree if desired. As the project expands, the included tool set will also need to expand.

Chapter 3

Background

3.1 RIT ME Design Courses

The last two undergraduate courses of the engineering curriculum at RIT are Senior Design I and II (also known as Capstone or Multi-Disciplinary Senior Design). Students have entered the course with a large amount of problem solving under their belts yet surprisingly little design experience. One of the largest differences between solving a problem and tackling a design challenge is right at the beginning. Problems tell you what's to be delivered at completion. Design, on the other hand, can be much more open ended. This "fuzzy front end" can leave students struggling and discouraged. Quite often, once they have a grasp on the scope of the project, they're off and running.

To address this traditional lack of design experience, modifications have been made to the Mechanical Engineering (ME) course sequence. There are currently four courses in the ME program at RIT that specifically deal with the design process. Cornerstone Design, implemented in the course sequence in academic year 2006-2007, is a required course taken in the third year. Design Project Management (DPM), initially offered in academic year 2003-2004, is a technical elective that is encouraged for future senior design project managers and can be taken in the fourth or fifth year. Senior Design (SD) I and II have historically been taken by students in their final year of undergraduate study. SD I and II have been taught with multi-disciplinary teams since academic year 2002-2003. Several examples throughout this thesis refer to these courses. Official RIT course descriptions

are provided in the glossary. Several papers [27, 28, 29, 30, 31, 32, 33] are available with regard to the implementation of the new course structure.

3.2 The EDGE Team Collaboration Environment

The Engineering Design Guide and Environment (EDGE)[1] was developed to help student teams collaborate on design projects. EDGE has been in development in various forms since 1997, but took a large step forward when Mr. Brian Sipos, an RIT BS/MS student in Electrical Engineering, incorporated several open source programs into the environment. The web site continues to undergo modifications as it adapts to address the aspects that students do no like. EDGE currently supports all four ME design courses. The benefits of the open source EDGE system are in it's ability to store all project information, along with version control while utilizing a simple-to-edit wiki interface. The web-based EDGE interface handles access control through an LDAP username and password. Each design team has it's own subversion repository for project document control. EDGE controls the user rights over viewing and editing pages. The underlying MySQL database for EDGE controls the project access and version control data management. All pages under EDGE maintain a consistent look and feel through CSS. Approximately six hundred students use the EDGE site annually through Cornerstone Design, Design Project Management, Senior Design I and II, and individual thesis projects. Each student team has a distinct project repository and web site to facilitate data sharing while providing a common architecture.

EDGE is also home to this thesis repository, P07898, which is similar to any project directory. It houses a web site location for providing information to the public as well as a private site for users with the appropriate permissions. The subversion repository allows all documents to be backed up, accessed from any location with secure internet access, and modifications to be tracked.

[1]The EDGE site, `https://edge.rit.edu`, is extensible and undergoing continual improvements. All statements regarding the capabilities of EDGE are current as of 2007.

3.3 The FACETS Design Process

A twelve-faceted design process was originally presented at the ASEE conference in 2001 by Hensel [34][p 2]. This process, as originally outlined is shown here.

Facet 1. Recognize and Quantify the Need

- Market Demand
- Assess competing solutions for the need
- Budgetary Parameters
- Develop formal Needs Statement and Statement of Work for Customer Approval

Facet 2. Concept Development

- Brainstorming Techniques
- Literature Review of alternatives
- Consensus Building

Facet 3. Feasibility Assessment

- Technical Feasibility
- Economic Feasibility
- Schedule Feasibility
- Performance Feasibility

Facet 4. Preliminary Design

- Preliminary Drawing Packages
- Assembly and Component Drawings
- Bill of Materials and Supplier Identification

Facet 5. Establishing Design Objectives and Criteria

- Performance Specifications
- Design and Implementation Specifications

- Evaluation criteria

Facet 6. Analysis of Problems & Synthesis into the Design

- Formal Problem Solving Method
- Assembly Drawing “Big Picture” integration
- Systems integration

Facet 7. Engineering Models - Simulation and/or Hardware

- Software simulations and CAD models
- Rapid Prototype and physical representations
- Proof of Concept Prototype

Facet 8. Detailed Design (DFx)

- Comprehensive Drawing Packages
- Line by Line review of codes and standards
- Design factors include: Safety, Manufacturability, Maintenance, Assembly, Manufacturing, Disassembly, Recycling, Quality

Facet 9. Production Planning and Tooling Design

- Pre-Production Prototype
- Flexible work cell design, die design, fixtures, tooling, automation
- Process diagrams and process flow sheets

Facet 10. Pilot Production

- Cell acquisition
- Operator training
- System commissioning

Facet 11. Transition to Commercial Production

- Capitalization
- Standardization and interchangeability

- Mass Customization

Facet 12. Product Stewardship

- Sales, Service, and Support
- Consumer feedback for continuous product improvement
- Product Line Migration
- Product Maintenance and Recall Procedures
- End of Life considerations

This twelve facet design process has evolved and facets have been reordered and renamed since the original introduction. The original and current twelve facets are shown in Table 3.1. The term "facet" is used rather than "step" or "phase" to suggest that the facets reflect issues that must be considered during the design process, and not necessarily the sequence in which those issues should be considered.

Table 3.1: FACETS Evolution

Facet	Original	Current
1	Recognize and Quantify the Need	Needs Assessment
2	Concept Development	Concept Development
3	Feasibility Assessment	Feasibility Assessment
4	Preliminary Design	Engineering Analysis
5	Establishing Design Objectives and Criteria	Tradeoff Assessment
6	Analysis of Problems & Synthesis into the Design	Preliminary Design Synthesis
7	Engineering Models	Engineering Models
8	Detailed Design (DFx)	DFX: Detailed Design
9	Production Planning and Tooling Design	Production Planning and Tooling Design
10	Pilot Production	Pilot Production
11	Transition to Commercial Production	Transition to Commercial Production
12	Product Stewardship	Product Stewardship

Needs Assessment It is very important to understand what the problem is prior to diving into tackling the solution. Needs Assessment helps the design team identify and quantify what the customer wants, whether these needs are explicitly stated or remain unstated. To gain a greater appreciation of what the customer is looking for, needs can be grouped or ranked. Affinity diagrams, objective trees, and pairwise comparisons are a few tools available to aid in the grouping and ranking. It should also be determined how the design team will measure if the product being developed has effectively satisfied the customer needs. Customer needs should be correlated to engineering specifications. House I of the HoQ is a helpful tool to show these correlations. Engineering specifications should have measures of performance associated with them. Once the customer needs and engineering specifications are defined, the functional requirements of the design are determined. Function trees help group these functional requirements.

Concept Development Once the functions are defined, ideas for solutions can be proposed. Ideas can come from several different techniques such as brainstorming, brainball or collaborative sketching. One important thing to remember is that concepts/ideas should be developed for each major function of the design, not just for the overall design solution itself. In doing this, a chart can be made showing the variety of concepts for each function. Various tools of expanding the conceptual design space include morphological charts and TRIZ [35]. Allowing team members to vote on concepts can create a ranked display of concepts, highlighting the best chance of a successful design combination.

Feasibility Assessment After developing a bank of ideas, it's very obvious that not all ideas can be fully investigated. To determine which ideas make the most sense to develop further, each function concept should be assessed on how it fulfills the needs of the customer. This assessment should use a numerical value to drive the best options. More critical needs can be given higher importance in the comparison. Design

concepts that do not satisfy the customer needs should not be further investigated. Following the CREST guideline, the design team needs to consider **C**onstraint feasibility, **R**esource feasibility, **E**conomic feasibility, **S**cope feasibility, and **T**echnical feasibility. Pugh's Method and Weighted Method are two tools to help in the feasibility assessment.

Engineering Analysis The majority of most undergraduate engineering curricula is targeted at addressing the skill set needed to perform engineering analysis. Finite element analysis, process modeling, and analytical behavior modeling are samples of analysis required to verify assumptions made earlier in Concept Development. The engineering sciences found in undergraduate curricula represent the engineering analysis toolbox. Mechanical engineering tools include thermodynamics, fluid dynamics, heat transfer, statics, mechanics, *etc.* Electrical engineering tools include circuits, electromagnetism, digital devices, control systems, *etc.*

Tradeoff Assessment It is very rare that everything desired in a design can coexist. A foam takeout container wants a highly secure, leak resistant closure and it should be easy to close [36]. These two requests can't both be achieved at the highest level. There needs to be some give and take, driven by the needs of the customer and subject to external constraints. Tools used in tradeoff assessment include engineering judgement, parametric studies, linear and nonlinear optimization.

Preliminary Design Synthesis The big picture is coming together as the detailed drawings, assembly drawing and the rest of the drawing package is pulled together. Suppliers should begin to be identified for key components. Tools include computer aided design (CAD), ECAD, bill of materials (BOM), parametric design, and corporate design practices.

Engineering Models From the desired design comes the prototype. This model can be used to further test, and show customers and marketing to verify that the needs are

being met. Models may be physical or virtual. Tools include wind tunnel models, clay mockups, prototypes, bench top tests and computer simulations.

DFx: Detailed Design Design For ... There are so many different possibilities here. Design for manufacturing, design for six sigma, design for assembly, design for intellectual property, design for safety, design for recovery. This takes the existing embodiment and pushes the design into thinking about other areas that will go into making and using the product.

Production Planning and Tooling Design How a product is made is just as important to consider in planning as what it's functions are. This includes fixture design, manufacturing process selection, and mold design. In order to save rework and lost time and money, its crucial to include this phase in the early design considerations. Tools include supply chain management, ISO 9000, and plant layouts.

Pilot Production A trial run of parts will help debug the rest of the process, as well as provide sample parts for testing, market studies, and customer approval. This should be a strong collaboration between manufacturing and design.

Transition to Commercial Production Commercial production can lead to new issues that may arise in handling, storing and shipping the product that's been made. Repetition could also affect the long term successfulness of the product.

Product Stewardship Product Stewardship entails dealing with sales, services, recalls, warranties, recycling, and continued support for a product after it's been delivered to the customer.

The FACETS process should not be viewed as a set of individual steps to be completed in a sequential order. Similar to a concurrent design model, many facets should be undertaken at the same time. If at any time the design team needs to go back and revisit earlier facets, it is encouraged to do so. Even after Product Stewardship, lessons learned need to be rolled

back into Needs Assessment to ensure that complications and faults are eliminated in the next round of design.

EDGE currently provides a body of design process knowledge for information related to the FACETS process. The FACETS process itself is not the single design process to be displayed to design teams utilizing EDGE, but a behind-the-scenes system for organizing information that is critical to all design processes. The FACETS process is not intended to replace existing design processes. As shown in **Concurrent Engineering Effectiveness** [8], each company needs to set up their own work process model. The FACETS process is intended to act as a translator and information parsing system between design processes. The Cornerstone Design course is using the Dym and Little [3] text with its design process and tools. Design Project Management is using the Ulrich and Eppinger [2] text with its design process and tools. Both of these texts address identifying customer needs and generating means to satisfy the design functions, but use different models and terms to describe those tools. EDGE has one project repository called "Cornerstone Design" which serves to map the Dym and Little process to FACETS. A second project repository called "Senior Design" maps the Ulrich and Eppinger process to FACETS. These separate views of the process all access the same information from the appropriate facet.

An analogy for the role of the FACETS process is in database software. Both MS Access and MySQL are commonly used databases, but the information contained within their files are not compatible with each other's program. Both programs can import and export comma separated value (.csv) files. If necessary, a user could write a native database file in an editor as simple as notepad using commas to drive the database, however, it's far more common to use .csv files as a means to move information previously viewed in an Access format to a MySQL format or vice versa. FACETS acts in a manner similar to the comma separated value file. Through it, the same information can be shared with many different design process interfaces. Further information on the FACETS process is provided through EDGE.

To demonstrate the ability of FACETS to act as an information parser between different design processes, Figure 3.1 shows the processes laid out in Section 2.2 mapped against the FACETS process.

Facets	Product Design and Development Ulrich, K., Eppinger, S. 2004	Engineering Design Dym, C., Little, P. 2004	The Engineering Design Process Ertas, A., Jones, C. 1996	Design for Electrical and Computer Engineers Ford, E., Coulston, C. 2005	Stage-Gate® Robert Cooper	Concurrent Engineering Effectiveness Fleischer, M., Liker, J. 1997
Needs Assessment	Planning	Client Statement (Need) / Problem Definition	Recognition of Need	Problem Identification / Research / Requirements Specification	Discovery / Scoping / Build Business Case	Identify Customer Needs
Concept Development	Concept Development	Conceptual Design	Conceptualization	Concept Generation	Development	Develop Concept
Feasibility Assessment			Feasibility Assessment			Get Market Reaction
Engineering Analysis		Preliminary Design	Preliminary Design	Design		Design Prototype
Tradeoff Assessment						
Preliminary Design Synthesis	System Level Design					
Engineering Models	Detail Design	Detailed Design	Detailed Design	Prototype and Construct		Build / Test Prototype
DFx: Detailed Design				System Integration	Testing and validation	Final Design
Production Planning and Tooling Design	Testing and Refinement	Design Communication	Production Planning and Tooling Design	System Test		Design Tooling, Make Tooling, Install Tooling, Purchase Parts
Pilot Production	Production Ramp up	Final Design (Fabrication Specs & Documentation)	Production	Delivery and Acceptance	Launch	Production Launch
Transition to Commercial Production						
Product Stewardship				Maintenance and Upgrade		

Figure 3.1: Design Process Comparison

Chapter 4

The Project Design

The thesis project developed is intended to be used initially by undergraduate engineering students. One unique benefit of implementing a database driven software system in a university environment is the breadth of the project topics. Information can be stored and recalled from diverse projects over many years. After a team identifies customer needs, they identify specifications. If a similar need has been included in a previous project, a script could be developed to suggest the specification based on prior work. If a project includes functionality that was included in a previous project, then the team could access concepts that were brainstormed at that time, including which concepts were chosen, and why. Having access to this wealth of knowledge should greatly increase the effectiveness of the senior design projects. It would also allow future generations to make improvements upon projects having full knowledge of the information that went into the decisions made.

Chapter 2 established that there are many well documented design processes that offer a guide for working on a new design and also many commonly accepted tools to help designers reach their design goal. Even with all this available information, without formal training on a design process, students in senior design struggled to get started on their projects[28]. In industry, commercially available software was created to help design groups tackle all stages of product development. Being only available commercially, evaluation of these alternative software package was based on screen shots and product literature. The software packages evaluated include:

- Artemis

- [http://aisc.com/Solution/3]

- Centric Software

- [http://www.centricsoftware.com/]

- MatrixOne

- [http://matrixone.com/]

- Optiva

- [http://www.formationsystems.com/products/]

- PD-Trak New Product Development Software

- [http://www.pd-trak.com/]

- PTC: Product Development System

- [http://www.ptc.com/products/product_development_system.htm]

The benefits of these commercial software packages include

- Project management tools

- Document sharing

- Professional user interface

- Tech support

The drawbacks include

- Cost

- Targeted at a particular industry

- Singular format based on the accepted design process

It appears that the majority of the software packages claiming to support new product development are mainly a combination of MS Project and a document repository. While these tools are critical to a team's success, they don't specifically address all the needs of a product development team.

The software developed for this thesis is open source and open architecture, meaning that it is free to use and modify to suit the design team's needs. It is hoped that the end deliverable software will be modifiable so that regardless of the design process employed by a company, the tools will be helpful and have a familiar feeling to them.

Through building the web site, similar patterns arose, suggesting an analogy to a tree. Looking deeper into the analogy revealed that a tree graph analogy was appropriate, and offered many levels of effectiveness. The analogy is explained below, followed by description of the underlying database supporting the project and the user interface scripts.

4.1 The Forest Analogy

For this analogy, the entire scope of Multi Disciplinary Senior Design (MSDS) at RIT is the forest and individual projects make the trees. Old projects still stand in the forest as they still have information to offer. As MDSD moves more into the project track format, these tracks can be shown as distinct types of trees in the forest. While some ideas for new projects may be planted, after a little nurturing, it may become apparent that these project saplings need to be pruned to make room for new projects. Concepts developed from one tree may be the seed that creates a new tree. Making all of the information in the forest easily accessible to those that need it will greatly increase the chances for successful trees to come. The project tree will be based on a project number. The roots of the tree are the people associated with making the tree grow. If a project identified appropriate focus areas, those optional focus areas would be handled like a trunk section (Figure 4.1). Each project has areas that need to be explored. These "branches" include customer needs, engineering specifications, design functions, and concepts. Because all projects need to explore these

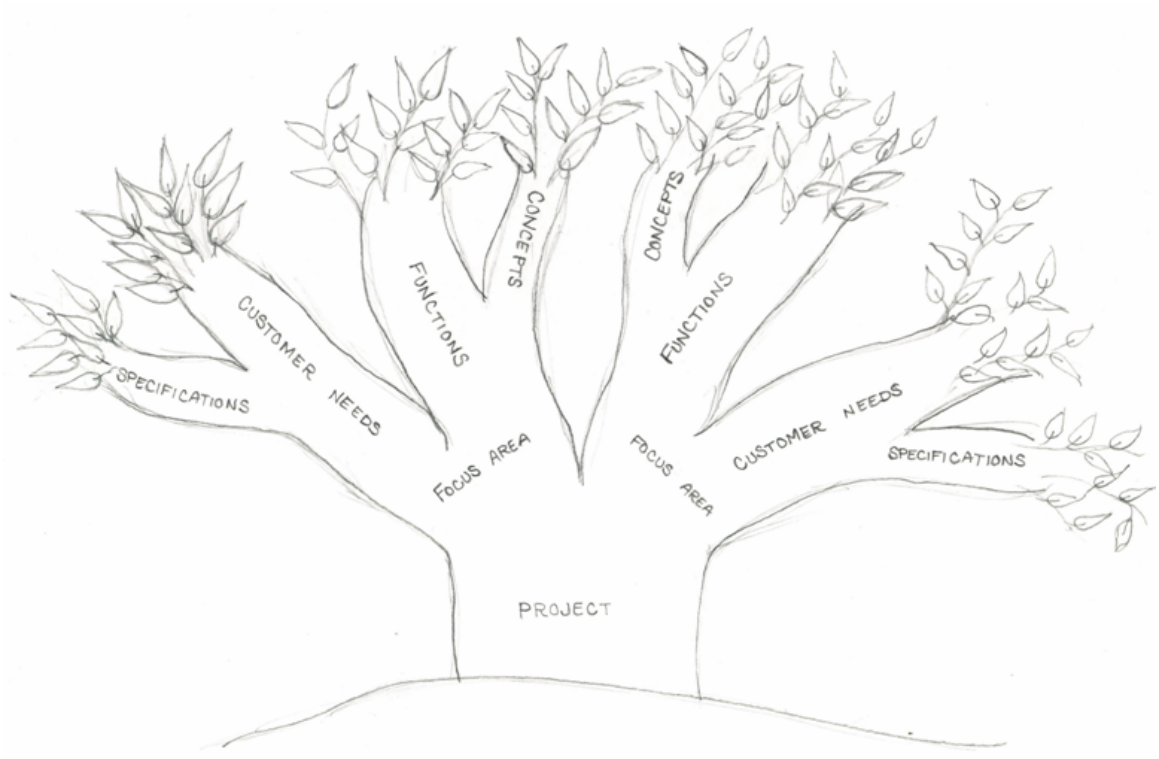


Figure 4.1: One Representation of a Project Tree With Two Focus Areas

areas, once the trunk section is created, these four branches are automatically created.

The project tree representation shows the specifications branch forming out of the customer needs branch. This implicitly suggests one example of information flow where specifications are determined based on customer needs. This applies similarly to the concepts branching off functions. These branches need to "fill out" before the tree is complete.

The leaf is the data. It's a piece of information associated with a branch. Leaves can be grouped together in clumps. They can also be arranged in the order on the branch depending on importance. While bare branches don't make a full tree, neither does a leaf pile. Only when leaves are arranged on the appropriate branches does the complete picture of the tree begin to take form.

To a student user, only trees of interest are available in EDGE. Students may have access to more than simply the tree that they are assigned to, due to the relevant leaves of

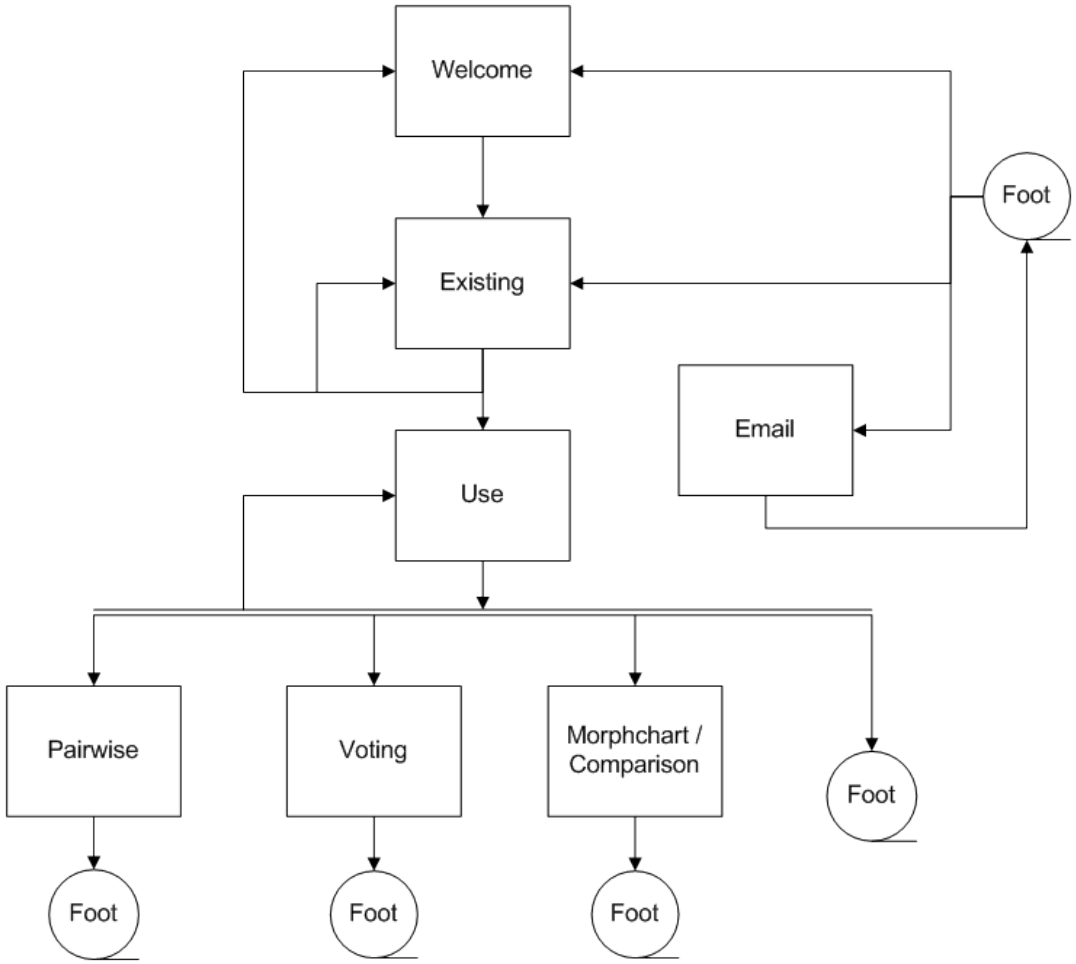


Figure 4.2: Overview Flow Chart

information among the branches of other trees. Administrators and faculty members may view the entire forest.

While this thesis project isn't implemented and available to all users in EDGE, it is treated as if it receives information from the main EDGE site. The flowchart shown in (Figure 4.2) shows the connections between the scripts of the project. After narrowing down the project scope, either through the welcome script or through EDGE, the first view that a user sees is a list of the branches in a project.

The first decision is whether to use an existing branch, delete a branch or create a new

branch. If the user presses the delete branch button, they are asked to confirm the deletion of that branch. If the user selects yes, the branch is deleted and the other branches of that tree are displayed. If the user selects no, then the branches of the tree are redisplayed without deleting a branch. If a user elects to create a new branch, the new branch or branches are created and displayed on the branch selection page. The user should now select that branch to use it. These decisions are all handled within the existing.php script.

The design from here presents recurring views to the user no matter what point in the process the user is at. The goal was to create a series of scripts that can be repetitively implemented, regardless of the task being tackled. There are software tools supporting gathering information, sorting information, comparing information and ranking information. This has been implemented at the early phases of the design process, and it is hoped that these same tools can continue to trickle down throughout the design process.

Upon selecting the branch to use, the user moves into the use.php script and may add and organize the leaves that belong on that branch. If there are no sorted leaves on the branch, the user is shown a list view of the leaves. Additional leaves can be created, or deleted. At any point the option exists to change branches. After leaves are entered in, they can be organized, similar to an objective tree. This defines where clumps form on the branch. Selected leaves can be sorted, unsorted, or deleted, or new leaves can be added. If a user wishes to order the importance of the leaves on a branch, a pairwise comparison is available through the pairwise.php script. Other mechanisms are available for ordering the importance of leaves, but scripts for alternative methods have not yet been developed.

Users should start adding leaves with Needs Assessment for the design process flow implemented in this thesis. Once the customer requirements begin to take shape, the team can move on to engineering specifications. When a user is entering in specifications, the unsorted view is similar, except the already created needs are shown in a list. The user can select which needs the specification addresses. The customer requirement leaves are on a different branch than the engineering specifications leaves, so these relationships are more like spider webs. These spider webs define the flow of information between leaves on

different branches. This flow of information also exists between functions and concepts. The design team should first focus on identifying the functions of the project, and then brainstorm concepts that are a means of achieving a single function. The morphchart.php and comparison.php scripts graphically display this information flow. The comparison chart can display these explicit relationships or use them to identify implicit relationships not thought of by the design team.

Pairwise comparisons help the team identify what the order of importance is among customer needs. Weighted voting, through the voting.php script, identifies which concept is the most appropriate method for achieving a function. These forms of ranking can be used to identify where the most critical aspects of design reside, and identify the most likely concepts for success.

These branches, leaves and relationships are all pieces of information stored in the database. This information is not dependent on a particular design process or a narrow set of tools. The database storing the information can support the ability to render the information in different design process views, or for different tools, given the appropriate user interface. In this sense, an affinity diagram created by *Team A* can be displayed as an objective tree for *Team B* allowing information to flow between tools used by different teams. Similarly, one engineer from *Team A* may prefer the affinity diagram graphical representation of the data, while another engineer on *Team A* may prefer an objective tree representation of the same data.

4.2 Database Design

The database server that stores all of the information used for the thesis is an SQL compliant database. MySQL was chosen because it is the same database server being used for `https://edge.rit.edu`. The features of the database server used herein are limited to those found in any SQL compliant database, such as MySQL [`http://mysql.com/`], PostgreSQL [`http://www.postgresql.org/`], or Oracle [`http://www.`

oracle.com/]. The information from the thesis project site is stored in a schema named test which is contained within the database server on a local drive. The information is version controlled through the SVN repository P07898 for this thesis on EDGE. This test schema will ultimately be incorporated into the FACETS schema which contains other information related to the design process. The EDGE schema manages access controls and information not related to the design process or design data itself.

The test schema uses eight tables, listed in Table 4.1. It is assumed that when included in the EDGE environment, these tables would also have access to other FACETS and EDGE schema tables, such as information about relationships between projects and project staffing. For instance, the table branch (Table 4.2) includes fields ProjectNumber and UserName. These fields should be foreign keys to information stored in the EDGE schema, not unique information. A foreign key identifies a column or a set of columns in one referencing table that refers to a column or set of columns in another referenced table. Adding in the information contained within the EDGE schema greatly increases the potential for maximizing the impact of the history that could be contained in the database. These tables are presented as proposed to be incorporated into the FACETS schema. The use of foreign keys did not become apparent until after development. When including the test schema tables in the FACETS schema, the foreign keys need to be created.

Table 4.1: Selected Tables From Schema test, Proposed for Incorporation Into the FACETS Schema

branch	Information entry session name
branchtype	Type options for creating sessions
branchfamily	Stores the flow between branches
leaf	Stores all data entered
leaffamily	Relationships between leaves
alpha	Comparison between leaves on a single branch
beta	Mapping between leaves on separate branches
gamma	Ranking of leaves with respect to leaves on a separate branch

The eight tables contained in the test schema store all of the information gathered by

the web "scripts" or "programs". There are some similarities between tables in the `test` schema. The first field of every table is `id`, which assigns an auto incremented integer, called the primary key, to each record. This value need never be seen by the user, and is only used internally to the software. The value of the `id` is to allow users to not worry about unique words or descriptive phrases. It is desirable to allow similar branch names and leaf names, as they create opportunities for incorporating a level of artificial intelligence into FACETS in the future. The last field of every table is a `TimeStamp` associated with the last time that the data in the record was modified. The `TimeStamp` is assigned in SQL using the intrinsic function `NOW()`. While the value of the `TimeStamp` isn't utilized in the current software, it has been included in all `EDGE` and `FACETS` tables and is available for use in future extensions. `EDGE` and `FACETS` also include a field called `LastEditor` to record the user name of the person responsible for the data created at the last time stamp. This field is not included in the current `test` schema tables, but should be considered for inclusion for consistency.

Efforts have been made to normalize the schema in some cases. Normalization is a sequence of steps by which a relational database model is both created and improved upon [37]. A normalized database reduces the occurrences of anomalies. There are several Normal Forms which are the steps contained within the normalization process. 1st Normal Form (1NF) removes repeating fields by creating a new table where the original and new table are linked together with a master-detail, one-to-many relationship. 2nd Normal Form (2NF) includes 1NF and creates a table where repeating values are removed to a new table. The result is a many-to-one relationship created between the original and the new tables. 3rd Normal Form (3NF) includes 1NF and 2NF and eliminates transitive dependencies. Transitive dependencies are relationships between two non-key fields. Examples of each of these Normal Forms are discussed as they are used in implemented tables. Going beyond 3NF, into 4th Normal Form, 5th Normal Form, or domain/key normal form (DKNF), offers higher protection from anomalies, but at a sacrifice to database performance [37]. The `test` schema is in 2NF. There are several transitive dependencies that keeps the schema

from being 3NF. The reasons behind these decisions are discussed below.

Each table in the test schema will be introduced in sequence. These eight tables store all of the pieces information and all of the relationships between the information. This schema is the backbone of the thesis project site.

branch

The table branch, Table 4.2, contains the descriptive information about each unique branch. The information for each record is stored in six fields: id, longer, type, ProjectNumber, UserName, and TimeStamp. The id, and TimeStamp fields

Table 4.2: branch Table Structure Proposed for Incorporation Into FACETS

Field	Type	Notes
id	Integer	Auto Increment, primary key
longer	VarChar	
type	Integer	foreign key
ProjectNumber	VarChar	foreign key
UserName	VarChar	optional, foreign key
TimeStamp	TimeStamp	NOW()

are the common fields discussed prior. The field longer is the branch name. If the branch is a top level branch, similar to the focus areas in Figure 4.1, then the branch name is entered in by the user and is a description of that focus area. Other branch names are based on the top level description and the type. The field type is a foreign key to branchtype.id (read, id field in the branchtype table). As mentioned above, the fields ProjectNumber and UserName should be foreign keys to information stored in the EDGE schema. The project number, the trunk of the tree, determines which branches are displayed to the user. The user name is an optional field that is not currently used in the site.

Placing the branch types in a separate table is an example of 2NF. The new table eliminates repeating values. The field branchtype is one issue keeping the schema from

being 3NF. The issue arises because the table has a field for the longer name and another field for the type, yet the type is commonly listed in the branch name. This violates 3NF because there could be an issue in which a user enters in a branch name of "Our Focus Area Functions" with a branch type that represents customer needs. It is acceptable to take this risk of anomaly at this time because the `longer` field may evolve to names that do not solely match the type.

`branchtypes`

The table `branchtype`, Table 4.3, associates a unique id with commonly occurring branch types. The table contains a field `shorter` and a field `longer`. The field `shorter`

Table 4.3: `branchtypes` Table Structure Proposed for Incorporation in `FACETS`

Field	Type	Notes
`id`	Integer	AutoIncrement, primary key
`shorter`	VarChar	
`longer`	VarChar	
`TimeStamp`	TimeStamp	NOW()

lists the name of the branch type, and the field `longer` gives a lengthier description. The branch types currently included are

1. Top (Focus area)
2. Customer Needs
3. (obsolete)
4. (obsolete)
5. Pairwise
6. Functions

7. Specifications

8. Concepts

As other branch types are identified, new entries may be added to the table `branchtype`. The type of the branch establishes the options that are displayed to user in the "scripts". As new branch types are identified, changes need to be made in the scripts to reflect the new options needed. Adding branch types is not done by the user through a script, but by the administrator through the schema.

The "top" type is used for the focus area of a project. The thesis, P07898, has two case studies, or two focus areas. There are two "top" type branches: Safe Beverage Container and BikeE. No leaf information is currently stored under the top branch, but there's no reason that it couldn't be in the future. The purpose of the "top" type is to identify optional focus areas within the project. The "customer needs", "functions", "specifications", and "concepts" branch types all hold leaves that were created using tools targeted at clarifying those aspects of the design. The "pairwise" type allows the user to choose a subset of another branch to perform a pairwise comparison tool on. Its purpose is to ease coding of the script for the pairwise comparison tool.

`branchfamily`

The table `branchfamily`, Table 4.4, establishes relationships between branches. The

Table 4.4: `branchfamily` Table Structure Proposed for Incorporation in `FACETS`

Field	Type	Notes
`id`	Integer	AutoIncrement, primary key
`Child`	Integer	foreign key
`Parent`	Integer	foreign key
`TimeStamp`	TimeStamp	NOW()

relationships between branches are what determine the flow of information. The flow chosen to be used in this thesis is based on concepts taught in the DPM and SD I courses.

This flow of information has the user identify the customer needs. Once the needs are identified, then engineering specifications are determined to express the needs. Functions also need to be determined by the user, after which the concepts are generated to fulfill the functions. To represent this flow in the database a script writes relationships into the table `branchfamily`. The flow of information as it is written in the database shown in Figure 4.3. The top branch will be a parent branch to the needs and functions branches. The

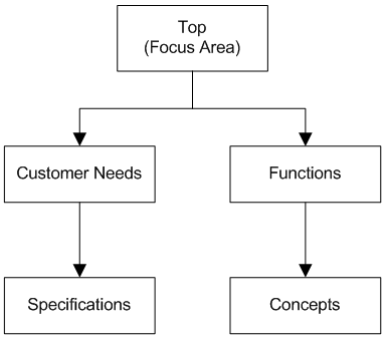


Figure 4.3: Flow of Information in `branchfamily` Table

needs branch is a parent to the specifications branch and the functions branch is a parent to the concepts branch. The `branchfamily` provides the relationships between two other existing pieces of information.

`leaf`

The `leaf` table, Table 4.5, keeps all of the pieces of information, and assigns them each a unique identifier. Each leaf can only be associated with one branch. This forces all leaves to be written within a context and maintains the history of where the information came from. To use a leaf in the context of a separate branch, the `longer` leaf name can be rewritten to the `leaf` table with a new `branch_id`. The `branch_id` relates to `branch.id`, the unique identifier of the branch that the leaf was created under. The reason for requiring that each leaf be associated with only one branch was to preserve the history of the decisions made by the design team. All of the information collected in the thesis project site is

Table 4.5: `leaf` Table Structure Proposed for Incorporation in `FACETS`

Field	Type	Notes
`id`	Integer	AutoIncrement, primary key
`branch_id`	Integer	foreign key
`longer`	VarChar	
`TimeStamp`	TimeStamp	NOW()

stored in the leaf table. The other tables, with the exception of the `branchtype` table, are methods of showing relationships between the leaves.

`leaffamily`

The table `leaffamily`, Table 4.6, stores relationships between leaves. When an objective tree or an affinity diagram is created, the `leaf.id` of the leaf that is the upper level leaf is written to a `Parent` field, with the leaf `leaf.id` below it being written to the `Child` field. The `branch_id` is the `branch.id` of the branch for both the parent leaf and the child leaf. This limitation is in place because currently objective trees and affinity diagrams are only conducted within the context of a single branch. Leaves can be a child in one record, and a parent in another record. In the `test` schema, parents can have multiple children and children can have multiple parents. The scripts, however, do not currently support assigning multiple parents to one child. A sibling relationship can inferred from leaves that share a common parent leaf. The `branch_id` is a constraint of both the `Child`

Table 4.6: `leaffamily` Table Structure Proposed for Incorporation in `FACETS`

Field	Type	Notes
`id`	Integer	AutoIncrement, primary key
`branch_id`	Integer	foreign key
`Child`	Integer	foreign key
`Parent`	Integer	foreign key
`TimeStamp`	TimeStamp	NOW()

and the Parent, not the record and could be determined from the leaf table. This inclusion means that the schema is not 3NF but including it here greatly aids the ease in coding.

The alpha, beta, and gamma tables are all used to store leaf relationships. It's not clear if three separate tables are necessary to store the relationships, but the relationships are all unique. The alpha table stores information regarding the relative importance of leaves. The beta table stores information regarding relationships between leaves on separate branches. The gamma table stores information regarding the appropriateness of a leaf with respect to another leaf. It is conceivable that these tables could be consolidated or expanded in the future.

alpha

The table alpha, Table 4.7, stores the data resulting from a pairwise comparison of two leaves on a common branch (Figure 4.4). A ranking of relative importance can be deter-

Table 4.7: alpha Table Structure Proposed for Incorporation in FACETS

Field	Type	Notes
id	Integer	AutoIncrement, primary key
branch_id	Integer	foreign key
leaf_1	Integer	foreign key
leaf_2	Integer	foreign key
Data	VarChar	optional
User	VarChar	optional, foreign key
TimeStamp	TimeStamp	NOW()

mined based on this information.

The branch_id typically has a type of pairwise, but it is not required. As before, the inclusion of the branch_id field is redundant but increases ease of coding. For the pairwise comparison, leaf_1 is the need that's displayed on the left for the user to select and leaf_2 is displayed on the right. The data field stores a 1, 0 or -1. If the user selected

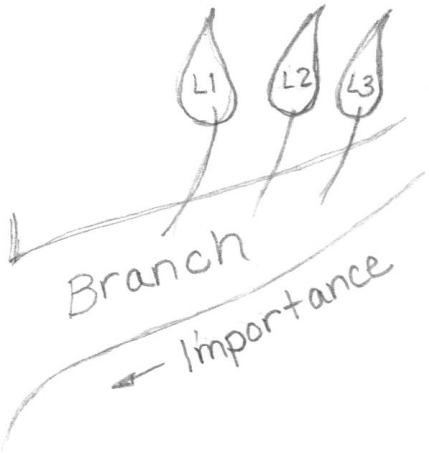


Figure 4.4: Leaves on a Common Branch in `alpha`

the left option, the `data` field stores a 1. So, the `data` field is the relative importance of `leaf_1` with respect to `leaf_2`. The `user` field is optional, and not currently being used. The `data` field is not limited to only using 1, 0 or -1. If it was determined that a strength of relationship quantifier was desired by the user, those modifications could be made in the scripts and supported without modification to the structure of the `alpha` table. It is also conceivable to use this table to store an assigned ranking by entering the same `leaf.id` in both fields `leaf_1` and `leaf_2`.

beta

The table `beta`, Table 4.8, is where the relationships between leaves on separate branches are defined (Figure 4.5). These relationships drive the connections within the HoQ and the morphological chart. In the analogy used in Section 4.1, these relationships form the spider web. The `beta` table stores the flow of information between leaves. When a user is entering in concepts, these concepts are created with respect to a function. That function is another leaf stored on a separate branch. When the new concept is entered in, a new leaf is added to the `leaf` table and then the relationship between the concept leaf and the

Table 4.8: `beta` Table Structure Proposed for Incorporation in `FACETS`

Field	Type	Notes
`id`	Integer	AutoIncrement, primary key
`branch_1`	Integer	foreign key
`leaf_1`	Integer	foreign key
`branch_2`	Integer	foreign key
`leaf_2`	Integer	foreign key
`Data`	VarChar	optional
`User`	VarCar	optional, proposed foreign key
`TimeStamp`	TimeStamp	NOW()

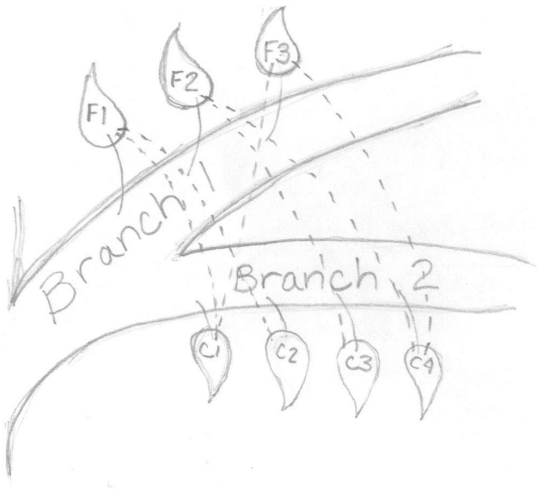


Figure 4.5: Relationships Between Leaves on Separate Branches in `beta` Table

function leaf is entered in the beta table. The leaf_2 and branch_2 are associated with the new concept leaf, and the leaf_1 and branch_1 are associated with the function leaf that the information flowed from. Currently, the existence of a record in the table indicates a relationship between leaf_1 and leaf_2. The data field is set up to have the potential to store information about the strength of relationship. Both the user and data fields are optional. The redundant branch_1 and branch_2 fields allow for ease in displaying the comparison table, which displays the information flow relationships.

gamma

The table gamma, Table 4.9, stores data resulting from the weighted voting tool. It stores the appropriateness of one leaf with respect to another leaf (Figure 4.6). The vote_item

Table 4.9: gamma Table Structure Proposed for Incorporation in FACETS

Field	Type	Notes
id	Integer	AutoIncrement, primary key
vote_item	Integer	foreign key
ref_item	Integer	foreign key
Data	VarChar	
User	VarChar	proposed foreign key
TimeStamp	TimeStamp	NOW()

is typically a leaf on a concept branch and the ref_item is typically a leaf on a function branch. The necessity for this table was determined once it was noticed that any given concept could be associated with multiple functions. A vote from weighted voting isn't associated purely with a leaf, it's associated with a leaf with respect to another leaf. The relationships between these leaves are created in the beta table and described as spider webs. The gamma table stores the votes for determining the best concept to solve a given function. This determines the thickness of the spider web connection created. Each record in the voting table is unique. If a user resubmits their vote on a particular function, it overwrites the record in the table. Since the leaf pairs being voted on are being retrieved

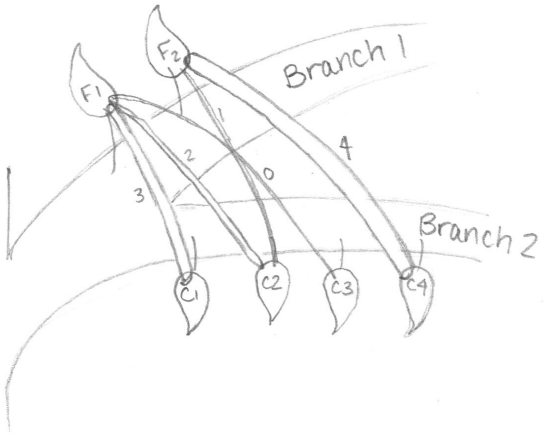


Figure 4.6: Appropriateness of Leaves With Respect to Another Leaf on Separate Branches in `gamma` Table

from the `beta` table, there was no coding benefit gained from including `branch_id` in this table.

The tables `alpha, beta,` and `gamma` all capture different information about the leaves and their relationships to other leaves. The `alpha` table captures the relative importance between two leaves. The `beta` table captures the relationship derived from the flow of information between leaves. The `gamma` table captures the appropriateness of a leaf in satisfying another leaf. It does appear that the `data` field in `beta` which represents the strength of relationship is very similar in functionality to the `data` field in `gamma` which represents appropriateness. It is also the case that there does not exist any leaf pairs in `gamma` that are not also leaf pairs in `beta`. It seems that a more efficient way to set up `beta` and `gamma` would be to use `beta` to create the relationship between leaves, and use `gamma` to create a record with `beta.id, data,` and `user` to store either the user name and vote or the strength of relationship entry. This modification would have a large impact on the scripts that display the comparison chart. It is also not yet known if `gamma` will always contain leaf pairs already defined in `beta`.

These are the eight tables behind the thesis project site. The table structure and names

evolved throughout development. Originally one family table existed that held both branch and leaf relationships. Combining the information that was used for different purposes made SQL queries cumbersome. Limitations in knowledge regarding database and script development has lead to redundancy in the schema. The schema as it exists today stores all the information and relationships for the tools provided for the Needs Assessment and Concept Development facets. This extensible schema can grow to accommodate additional tools and more facets as they are developed.

All information contained in the schema as of July 12th, 2007 has been exported in .csv files. These files are located in the P07898 project repository in EDGE. The url is `https://edge.rit.edu/content/P07898/public/schema`.

4.3 Scripts

The scripts provide the user the ability to write and read information contained within the `test` schema without needing direct access to the database or ever being aware that data is being stored in a database. The flowchart presented in Section 4.1 (Figure 4.2) shows an overview of the main pages of the project. Each proposed individual page will be examined herein. A summary table is provided at the end. The bottom of every page has two buttons. The first button returns the user back to the starting page, welcome.php. A second button allows the user to email the webmaster using their default mail editor.

Words shown in the san-serif font describe PHP functions or variables. Words shown in the `typewriter` font refer to database tables or fields. Functions that begin with Tree are stored in TreeFunction.Library.php which all pages have access to. Branches are referred to as sessions in the user interface. The user does not see any of the tree analogy terms. They are used purely for describing the schema structure.

4.3.1 Welcome

Welcome (Figure 4.7) is the starting point and lets the user narrow down the branches in the schema and only display those with a `ProjectNumber` similar to the project search term. Welcome prompts the user for information on which project to display branches for. The project information is passed to the existing.php. When the thesis tools are incorporated into the EDGE site, there will no longer be a need for welcome. The functionality of project access control should be accomplished through EDGE.

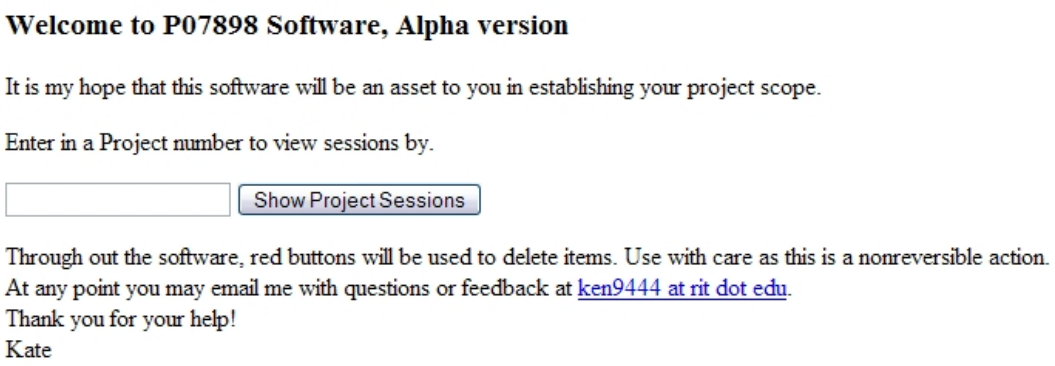


Figure 4.7: welcome.php Page

4.3.2 Existing

Existing displays the the branches with a `ProjectNumber` like the search term sent by welcome and provides users with several options for using the branches (Figure 4.8). The `test` schema tables `branch` and `branchfamily` are rendered and written through this page. In rendering existing branches, branches are shown according to the `branchfamily` relationships with the child branch being display below and indented from the parent. The options presented for the user are "Use the selected session", "Delete Session", or "Create a new session". These options are shown in the flowchart shown in Figure 4.9. As always, the user has the option to return to welcome or email the webmaster.

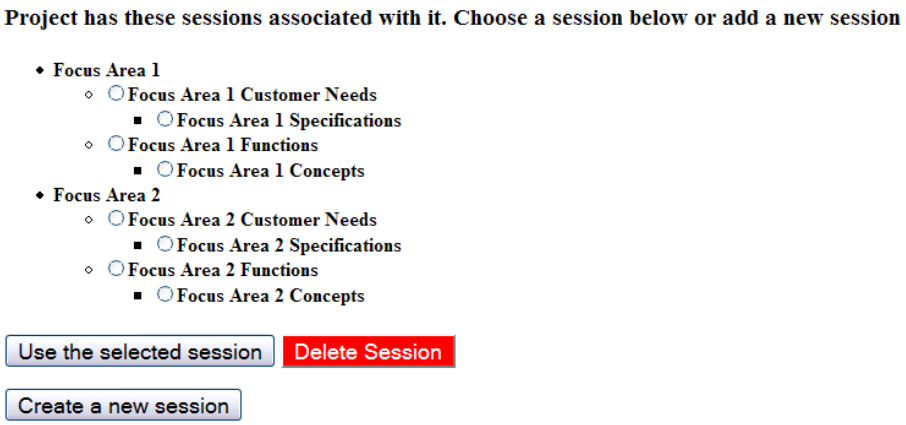


Figure 4.8: existing.php Page

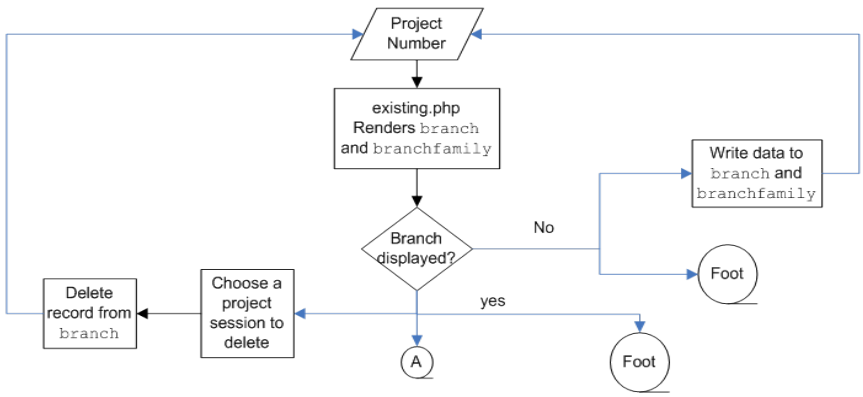


Figure 4.9: existing.php Page Flow Chart

To either use a branch or delete a branch, the radio button next to the desired branch needs to be selected. Top level branches are not currently selectable, since leaves should only be associated with a branch that describes the purpose of the leaf. This means that the top level branches are not able to be used or deleted using the user interface. One line in the TreeSession function is commented out, removing the functionality of a select option in front of the top level branch. The user could be allowed to select a top level branch if the line was uncommented.

When a branch is selected for deletion, the `branch_id` is passed to use. In this case, use is only a confirmation page (Figure 4.10). Only if the user selects "Yes, Delete the

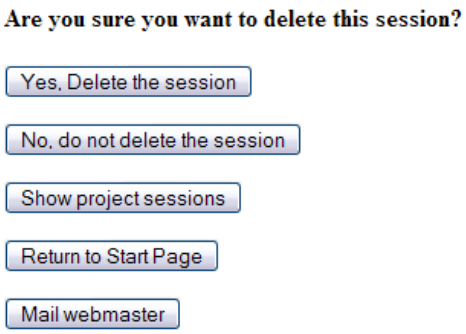


Figure 4.10: Delete Confirmation Screen Capture on use.php

session" will the branch be deleted from the schema. After selecting "Yes, Delete the session", the `branch_id` is sent back into existing with a variable delete. When existing confirms the variable delete has been set, the function TreeDeleteSession removes the branch from the table `branch`. The foreign key cascades the delete action, removing the associated leaves and relationships. After the branch has been deleted, or if the user selects "No, do not delete the session", existing displays all remaining branches on the project tree.

If the user selects a branch and chooses "Create a new session", the new branch created will be added to the table `branchfamily` as a child to the branch that was selected. If no existing branch is selected then the new branch will not have a record in `branchfamily`.

The page newsession is used for collecting the information to be written to `branch` (Figure 4.11). Newsession does not read or write information from any of the tables, it just passes information back to existing. Currently newsession doesn't offer the choice

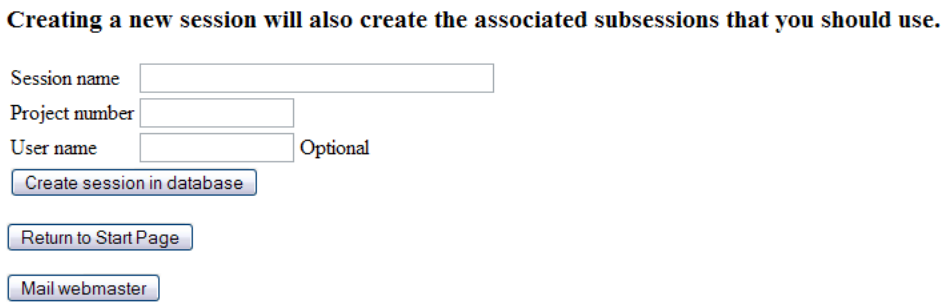


Figure 4.11: Information to be Filled out for Creating a New Branch in newsession.php

of branch type to the user. All new branches are created by default as top level branches. To give the user the option of setting a type, newsession should be modified to display a pull down list of the types from `branchtype`. All information collected gets passed back to existing when the user selects "Create session in database".

When existing loads coming from newssession, there is a variable identifier letting existing know to write the new branch to the table `branch`. The function TreeSaveTop writes the record into the table `branch`. If the `branch.type` is 1, then the function TreeSessionSetup writes the Customer Needs, Specifications, Functions and Concepts branches to the table `branch`, and writes the flow of information relationships to table `branchfamily`. After the new branches have been written, existing renders all of the branches associated with the project.

"Use the selected session" sends the `branch_id` of the selected branch to use.

4.3.3 Use

Use is where all the leaves are entered and written to the table `leaf`. It is also where relationships between leaves are defined and written to `leaffamily`. A `branch_id` is

passed into use to associate the leaves with. Use offers two main views: unsorted and sorted (Figure 4.12). The unsorted view renders all leaves with the matching `branch_id`,

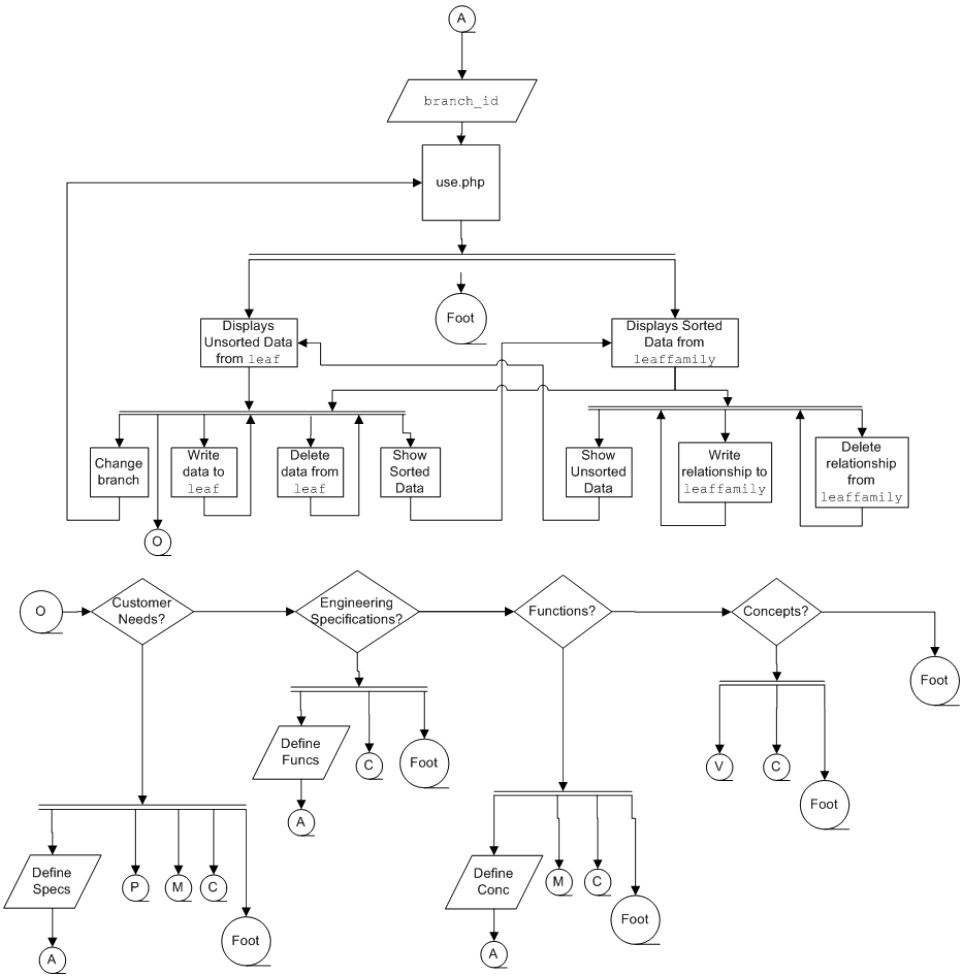


Figure 4.12: Flowchart of use.php

allows the user to enter in new leaves or delete existing leaves, and acts as a traffic controller, providing a series of links to other tools based on the `branch.type` (Figure 4.13). The sorted view renders all leaves according to the relationships stored in `leaffamily`, allows the user to sort or unsort existing leaves, creates new leaves or deletes existing leaves and provides the traffic controller (Figure 4.14). In the case of specifications and concepts, the leaves are entered with a relationship to one or more needs or functions, respectively.

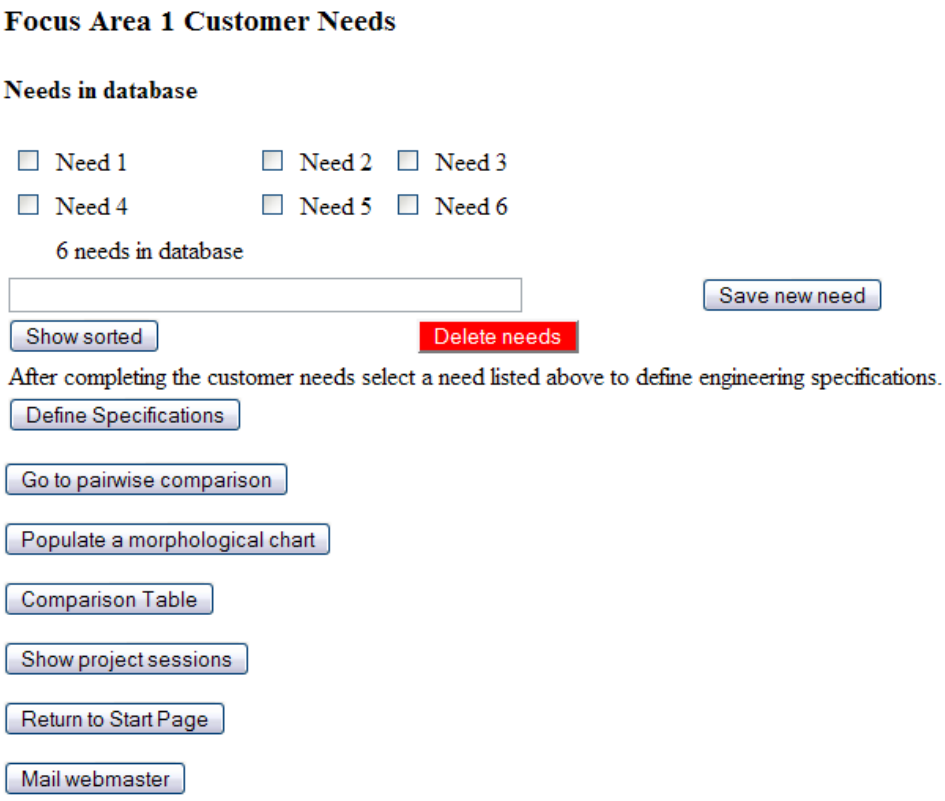


Figure 4.13: Screen Capture of Unsorted View of use.php

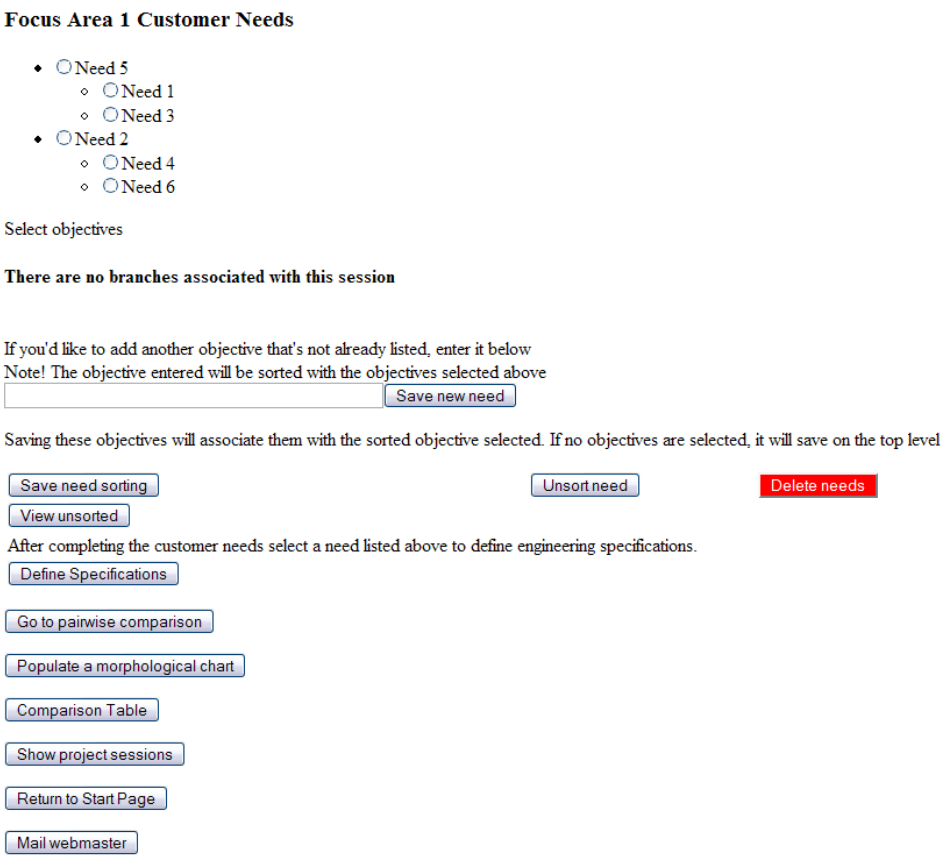


Figure 4.14: Screen Capture of Sorted View of use.php

When these leaves are created, in addition to writing the leaf to table `leaf`, the relationship is written to the table `beta`.

unsorted

The unsorted view of use is a location for entering raw data, such as data resulting from brainball (Section 2.3.2) or brainstorming(Section 2.3.3). Multiple users can be entering information simultaneously. Information is entered into the text box and written to `leaf` when "Save new need" is clicked. This leaf is immediately displayed with the other leaves already in the table in the order that it was entered in. All leaves that are checked are deleted when the user hits the "Delete need" button. There is not currently confirmation on the delete leaf function, although it is desirable to implement it. All four branch types allow the user to create a leaf without relating it to another leaf. It is preferred that specifications be related to needs and concepts be related to functions. When viewing the unsorted view of specifications, a list of the previously defined needs is shown. The user may select one or more needs that the specification being defined describes. This relationship is stored in the `beta` table. A list of previously defined functions is displayed for selection when generating concepts. This user interface could be improved by allowing a user the option of selecting a single function and then all subsequent concepts entered would be associated with that one function, rather than having the user reselect the function for every concept, which could lead to error.

sorted

The sorted view displays grouped leaves according to the `leaffamily` table. This is representative of an objective tree (Section 2.3.8), an affinity diagram (Section 2.3.1), or a function-means tree (Section 2.3.5). The user can still create leaves in the sorted view. The top portion of the screen shows the sorted leaves with radio select buttons. Below that all unsorted leaves are shown with check boxes. By selected a sorted leaf and one or more unsorted leaves and choosing "Save need sorting", the user can group the unsorted leaves

under the sorted leaf. The text box shown allows a user to create new leaves using "Save new need" as the grouping reveals missing information. Leaves can also be deleted using "Delete needs" or unsorted by selecting "Unsort need".

Traffic Controller

The traffic controller in use offers the user a number of links to tools based on the branch type being used. TreeUnSortedOptions and TreeSortedOptions contain the drivers that determine which tools are shown when. All branch types have the same bottom options of "Show project sessions", "Return to Start Page" or "Mail webmaster". If the user is looking at the unsorted view, all branch types have an option to see the sorted view, and vice versa.

The additional links offered under the Customer Needs branch type include "Define Specifications", "Go to pairwise comparison", "Populate a morphological chart", and "Comparison Table". "Define Specifications" reloads use using the `branch_id` of the specifications type branch. The other three options send the `branch_id` to pairwise, morphchart, and comparison respectively. The Specifications branch type options are "Establish functions" and "Comparison Table". Similar to "Define Specifications", "Establish functions" reloads use using the `branch_id` of the functions type branch. The Functions branch type options are "Brainstorm concepts", "Populate a morphological chart", and "Comparison Table". "Brainstorm concepts" reloads use using the `branch_id` of the concepts type branch. The Concepts branch type options are "Vote or Rank", which sends the `branch_id` to voting and "Comparison Table". These options are shown in the flowchart in Figure 4.12.

4.3.4 Pairwise

Pairwise populates and renders a pairwise comparison chart (Section 2.3.9) based on the `branch_id` sent from use (Figure 4.15). If the branch type is pairwise, the page displays the matrix and comparison buttons based on information stored in the table `alpha`. If the branch type is not pairwise, the customer need leaves are displayed with check boxes to

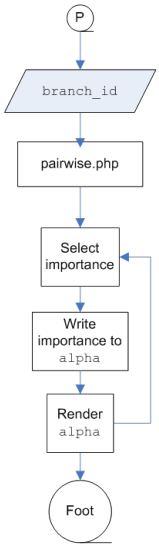


Figure 4.15: Flowchart for pairwise.php Page

allow the user to choose "Use the selected options" or "Use all the options" to perform the pairwise comparison on (Figure 4.16). In the case that the user selects "Use the se-

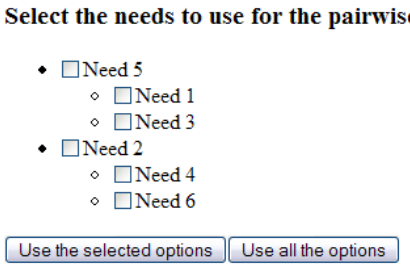


Figure 4.16: Branch Selection for pairwise.php

lected options", a new branch is written to table `branch`. A relationship is also written to `branchfamily` where the `branch_id` that is passed to pairwise is the `parent` and the newly created branch is the `child`. The leaves that were checked are rewritten to the table `leaf` with the new `branch_id` using TreeSaveGroup. The redundancy in writing information to `leaf` was decided for ease in coding. It would be desirable to be able to select

the leaves to perform the pairwise on and enter those leaves into the table `alpha` without needing to create a new branch. In the case that the user selects "Use all the options", pairwise uses the leaves on the existing branch without creating a new branch.

Once the leaves are selected, the user is shown two leaves and asked to determine which leaf is more critical to the customer. As shown in Figure 4.17, the user is offered 3 buttons; "Need 1", "Equally Important", or "Need 2". "Need 1" and "Need 2" cycle through all of the leaves to completely populate the chart. Clicking the left leaf button writes a record with the `branch_id`, the `leaf.id` of the left leaf to the field `leaf_1`, the `leaf.id` of the right leaf to the field `leaf_2`, and a 1 for the field `Data` to the table `alpha`. Clicking the right leaf button writes the same information, except a -1 for the field `Data`. Clicking on the "Equally Important" button writes a 0 to the field `Data`. All of the individual comparison records will be saved and any modifications will update the record. TreePairwiseRelate renders the information in `alpha` into a matrix and TreePairwiseRank compiles the total for each leaf to be displayed at the bottom of the matrix. Clicking on "Show needs in rank order" uses TreePairwiseResults to display a sorted list of the leaves (Figure 4.18).

4.3.5 Morphchart and Comparison

Morphchart and comparison are two different tools for displaying the relationships between leaves on separate branches. This is done by rendering information stored in the table `beta`. The morphological chart (Section 2.3.7) displays the concepts associated with functions whose `branch_id` is passed to morphchart from use (Figure 4.19). The purpose of this tool is to give the user a one look summary of the concepts associated with all of the functions of the design (Figure 4.20). The concepts down a column are explicit relationships defined in `beta`. The information gained from using a morphological chart is identifying previous unthought of concept combinations. In this way, these concept combinations become implicit relationships, displayed in the user interface, but not written in the schema. Morphchart does not write any information to the schema.

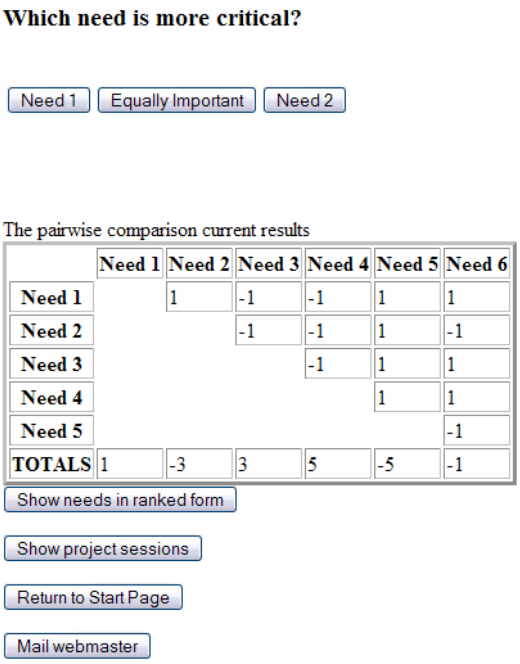


Figure 4.17: Matrix From pairwise.php

Comparison also renders information contained in the table `beta` based on the `branch_id` passed from use (Figure 4.21). Comparison doesn't use just the branch that is passed in, but offers the branches with `branchfamily` relationships under one top branch. Any branch can be displayed on either axis (Figure 4.22) . The relationships between leaves on different branches stored in `beta` are displayed in the middle, similar to a House of Quality style matrix (Section 2.3.6). An 'X' represents that a relationship exists, either explicitly as a record in `beta` or implicitly as a connection of explicit relationships. A numerical value stored as `data` in table `beta` indicates the strength of relationship. Displaying a branch against itself shows the tradeoff relationships represented by the roof in a House of Quality. Currently comparison only renders `beta` information. Provided the appropriate interface, the user should be able to use comparison to edit or add relationships to `beta` directly from the matrix.

Morphchart and comparison are the only tools that display implicit information. All

Which need is more critical?

Thank you for completing this Pairwise Comparison!

The pairwise comparison current results

	Need 1	Need 2	Need 3	Need 4	Need 5	Need 6
Need 1		1	-1	-1	1	1
Need 2			-1	-1	1	-1
Need 3				-1	1	1
Need 4					1	1
Need 5						-1
TOTALS	1	-3	3	5	-5	-1

Show needs in ranked form

Need 4 5

Need 3 3

Need 1 1

Need 6 -1

Need 2 -3

Need 5 -5

Show project sessions

Return to Start Page

Mail webmaster

Figure 4.18: Totals for pairwise.php

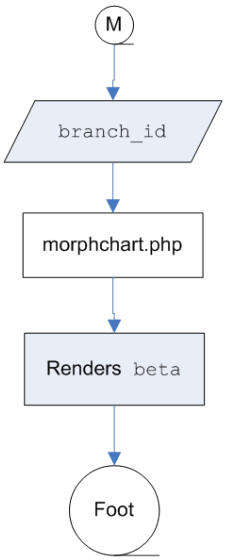
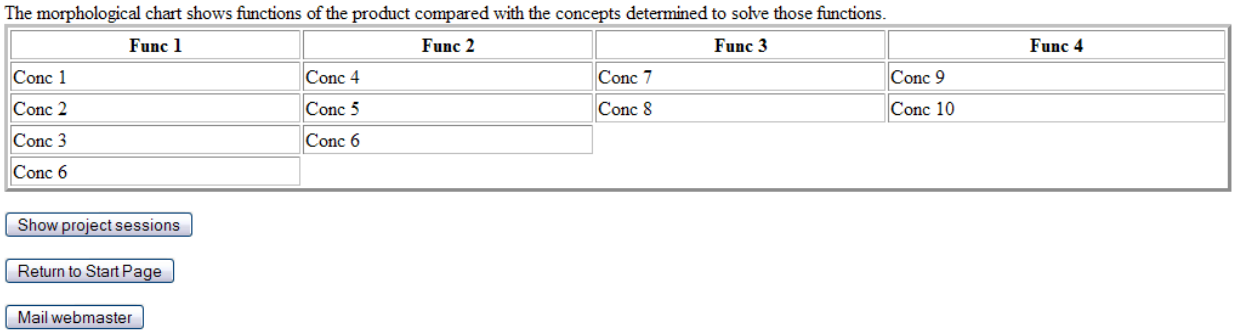


Figure 4.19: Flowchart for morphchart.php Page



The morphological chart shows functions of the product compared with the concepts determined to solve those functions.

Func 1	Func 2	Func 3	Func 4
Conc 1	Conc 4	Conc 7	Conc 9
Conc 2	Conc 5	Conc 8	Conc 10
Conc 3	Conc 6		
Conc 6			

Show project sessions

Return to Start Page

Mail webmaster

Figure 4.20: Screen Capture of morphchart.php Page

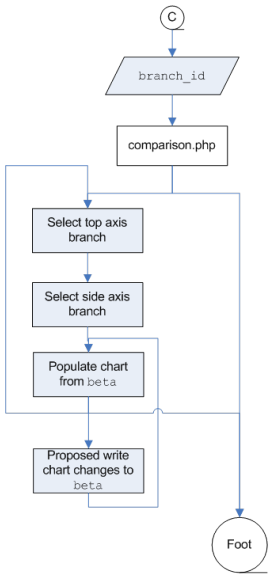


Figure 4.21: Flowchart for comparison.php Page

Choose a Session to display across the top.

Focus Area 1 Specifications

Choose a Session to display down the side.

Focus Area 1 Customer Needs

Populate Chart

The comparison chart displays a matrix comparison between two selected sessions.

	Spec 1	Spec 2	Spec 3	Spec 4	Spec 5	Spec 6	Spec 7
Need 1	X						
Need 2		X	X				
Need 3				X	X		
Need 4				X	X		
Need 5						X	X
Need 6							X

Show project sessions

Return to Start Page

Mail webmaster

Figure 4.22: Screen Capture of comparison.php Page

of the other tools are simply methods of writing and rendering explicit user input. This implicit information offers the user suggestions such as concept combinations and previously unidentified relationships. It is here in the implicit relationships that the flow of information between tools can be shown.

4.3.6 Voting

Voting stores weighted votes (Section 2.3.10) by a single user on the appropriateness of a concept with respect to a selected function and displays the total vote (Figure 4.23). Voting

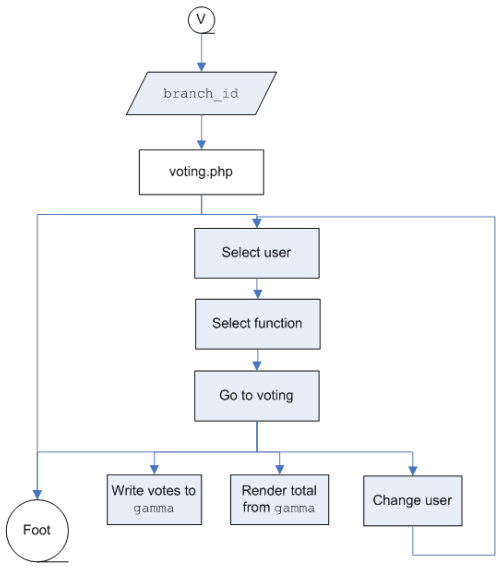


Figure 4.23: Flowchart for voting.php Page

both writes and renders table `gamma`, as well as using information contained in `branch, branchfamily, leaf, and beta`. Use sends the `branch_id` of a branch with the type "concepts". Voting is currently offered only to weigh the appropriateness of a number of concepts with respect to a function. To apply this script to another branch type, modifications would need to be made to TreeVotingFunction, which finds the function type `branch_id` and returns an array of the leaf id's and names. Before voting can begin,

the first step is to enter a user name and select the function to complete the voting on. When included in the EDGE site, the user name field should be determined automatically. After a number of votes per person is decided, each user can distribute votes among the displayed concepts (Figure 4.24). Selecting "Enter voting" writes the vote and user name to the

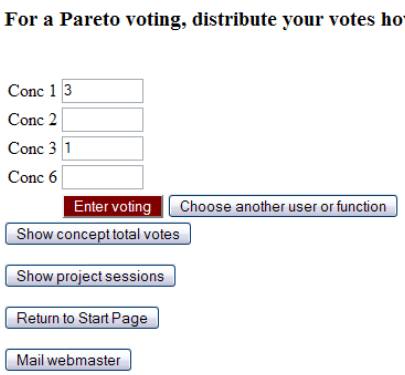


Figure 4.24: Screen Capture of Entry to voting.php Page

`gamma` table. The vote is rendered, and can be modified if desired, or another function or user name can be selected. The next time the user attempts to vote for concepts associated with the same function, the votes in the database will be displayed, and can be modified if desired. Selecting "Show concept total votes" displays a total of all of the votes to show the cumulative votes of the team (Figure 4.25).

As a side note, voting is not performed on any pair of leaves that isn't already established as a leaf pair in the `beta` table. It is not clear that this will always be the case as the tool set expands. If it was the case, then it might be more efficient to write the `gamma` table with `beta.id` rather than restating the leaf pair.

The information gained from this tool could conceivably be used to order the concepts in morphchart, resulting in the most appropriate concepts being displayed at the top of the chart, or displayed in comparison as a means of indicating the strength of relationship.

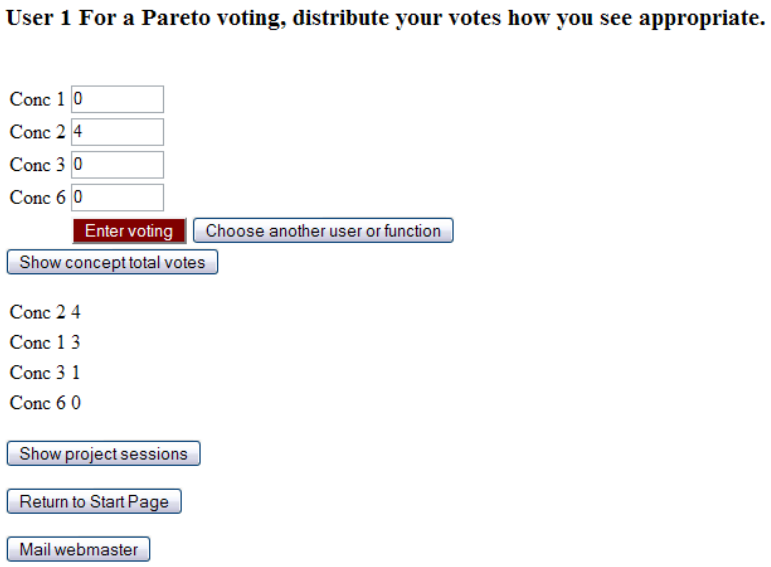


Figure 4.25: Screen Capture of voting.php With Totals

4.3.7 Summary

The scripts provided are an example of a user interface to the underlying schema. The relationships of the scripts to the schema tables that they manipulate are shown in Figure 4.26. The user interface will be the only means that a design team has of interacting with the schema, making it a very important aspect of the design. It is expected that this user interface will continue to evolve after implementation to provide the most efficient and understandable work platform.

All of the .php files used in creating the thesis project site have been placed in the P07898 project repository in EDGE. The url is `https://edge.rit.edu/content/P07898/public/scripts`.

Page name	Passed in	Passed out	Tables accessed	Tables always written to	Tables sometimes written to
welcome	nothing	project search keyword			
existing	project search term, project number	branch number, project number	branch, branchfamily	branch, branchfamily	
use	branch number	branch number, project number	leaf, leaffamily, branch	leaf, leaffamily	beta
pairwise	branch number	branch number, project number	leaf, leaffamily, alpha	alpha	leaf, branch
morphchart/ comparison	branch number	branch number, project number	leaf, branch, beta	proposed beta	
voting	branch number	branch number, project number	leaf, branch, beta, gamma	gamma	

Figure 4.26: Summary Table of Scripts

Chapter 5

Case Studies

Two case studies were performed using information supplied from text books to test the ability to translate information into a useable form. Several of the images in this section were included in Section 2.3 and are redisplayed here for convenience.

5.1 Safe Beverage Container

The textbook **Engineering Design** by Dym and Little [3] is being used for Cornerstone Design. This book was chosen for a case study because it follows several projects through various phases. The first case study is the Save Beverage Container from **Engineering Design**[p 58] having the problem statement

> "Design a safe method of packaging and distributing our new children's juice product that preserves the taste and establishes brand identity to promote sales to middle-income parents."

To start the case study, a new top level branch was created called "Safe Beverage Container". The associated project number is DYM. Once the new top level branch was created, the four supporting branches were also created; "Safe Beverage Container Customer Needs", "Safe Beverage Container Specifications", "Safe Beverage Container Functions" and "Safe Beverage Container Concepts".

Following a customer interview, what the client wanted from the design was cleared up and a list of the desired attributes was compiled. The attributes are shown in Figure 5.1. Attributes shown italicized are constraints, not objectives. By comparison, the attributes

Safe	→	DIRECTLY IMPORTANT
Perceived as Safe	→	Appeals to Parents
Inexpensive to Produce	→	Permits Marketing Flexibility
Permits Marketing Flexibility	→	Promotes Sales
Chemically Inert	→	*Constraint* on Safe
Distinctive Appearance	→	Generates Brand Identity
Environmentally Benign	→	Safe
Environmentally Benign	→	Appeals to Parents
Preserves Taste	→	Promotes Sales
Easy for Kids to Use	→	Appeals to Parents
Resists Range of Temperatures	→	Durable for Shipment
Resists Forces and Shocks	→	Durable for Shipment
Easy to Distribute	→	Promotes Sales
Durable for Shipment	→	Easy to Distribute
Easy to Open	→	Easy for Kids to Use
Hard to Spill	→	Easy for Kids to Use
Appeals to Parents	→	*Promotes Sales*
Chemically Inert	→	*Constraint* on Preserves Taste
No Sharp Edges	→	*Constraint* on Safe
Generates Brand Identity	→	Promotes Sales
Promote Sales	→	DIRECTLY IMPORTANT

Figure 5.1: Attribute List for the Safe Beverage Container From **Engineering Design** by Dym and Little [3][p 57]

are shown entered into the project website in Figure 5.2. The use.php script is used to write these attributes as leaves in the `leaf` table, associated with a branch called "Safe Beverage Container Customer Needs".

Attributes may be grouped using an affinity diagram or an objective tree. The safe beverage container objective tree is shown in Figure 5.3. Figure 5.4 displays the attributes grouped in an objective tree using sorted view of the use.php script. The relationships are stored in the `leaffamily` table. The differences between the Figure 5.3 and Figure 5.4 are purely graphical, the content and information are identical. It is necessary to develop a set of graphical renderers to display the information in an objective tree format.

Safe Beverage Container Customer Needs

Needs in database

☐ Safe	☐ Environmentally benign	☐ Promotes sales
☐ Easy to distribute	☐ Preserves taste	☐ Appeals to parents
☐ Permits marketing flexibility	☐ Generates brand identity	☐ Durable for shipment
☐ Easy for kids to use	☐ Perceived as safe	☐ Inexpensive to produce
☐ Distinctive appearance	☐ Resists Forces and Shocks	☐ Resists Range of Temperatures
☐ Easy to open	☐ Hard to spill	

Figure 5.2: Attribute List for the Safe Beverage Container as Displayed in Project Website

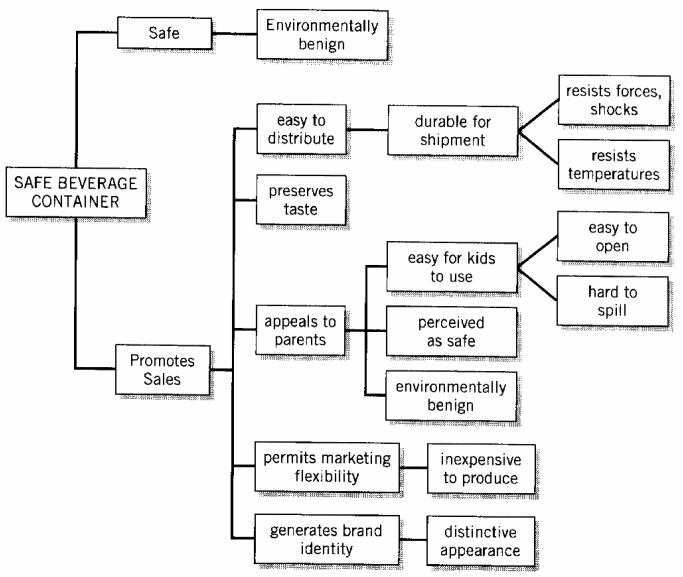


Figure 5.3: Objective Tree for the Safe Beverage Container From **Engineering Design** by Dym and Little [3][p 58]

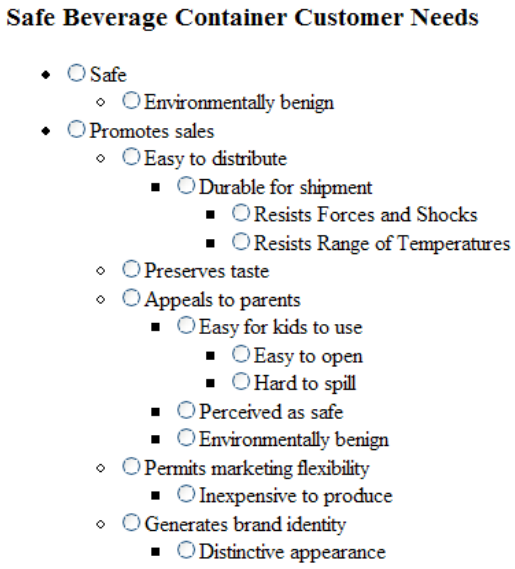


Figure 5.4: Objective Tree for the Safe Beverage Container as Displayed in Project Website

Dym and Little do not perform a pairwise comparison on all of the determined needs, but rather on a subset of those needs, only comparing objectives that are located at a similar level in the objective tree. The book also shows two different pairwise comparisons (Figure 5.5) to show that the priorities vary depending on who the client is. The project website only stores one pairwise. The pairwise shown in Figure 5.6 represents the weighted objectives for the company *Bringing Juice Into Children* (BJIC). To perform a pairwise on a subset of the needs, a new branch is written to the `branch` table called "Safe Beverage Container Customer Needs Pairwise". This subset of leaves is rewritten to the `leaf` table with the new `branch_id`. The selection of the subset of needs is handled through pairwise.php, as is the rendering and writing of the pairwise comparison tool. The individual comparisons are stored in the `alpha` table. The difference between the Figure 5.5 and Figure 5.6 is perhaps most evident in the totals. Dym and Little utilize the entire matrix for the comparison, tallying the number of instances that an objective is chosen to be more important. The website only displays the upper triangle of the matrix. This upper triangle provides complete coverage of the comparison, but yields a different total value than the

Goals	Environ. Benign	Easy to Distribute	Preserve Taste	Appeals to Parents	Market Flexibility	Brand ID	Score
Environ. Benign	••••	0	0	0	0	0	0
Easy to Distribute	1	••••	1	1	1	0	4
Preserve Taste	1	0	••••	0	0	0	1
Appeals to Parents	1	0	1	••••	0	0	2
Market Flexibility	1	0	1	1	••••	0	3
Brand ID	1	1	1	1	1	••••	5

(a) GRAFT's weighted objectives

Goals	Environ. Benign	Easy to Distribute	Preserve Taste	Appeals to Parents	Market Flexibility	Brand ID	Score
Environ. Benign	••••	1	1	1	1	1	5
Easy to Distribute	0	••••	0	0	1	0	1
Preserve Taste	0	1	••••	1	1	1	4
Appeals to Parents	0	1	0	••••	1	1	3
Market Flexibility	0	0	0	0	••••	0	0
Brand ID	0	1	0	0	1	••••	2

(b) BJIC's weighted objectives

Figure 5.5: Pairwise Comparisons From **Engineering Design** by Dym and Little [3][p 65]

	Environmentally benign	Easy to distribute	Preserves taste	Appeals to parents	Permits marketing flexibility	Generates brand identity
Environmentally benign		1	1	1	1	1
Easy to distribute			-1	-1	1	-1
Preserves taste				1	1	1
Appeals to parents					1	1
Permits marketing flexibility						-1
TOTALS	5	-3	3	1	-5	-1

Figure 5.6: Pairwise Comparison for the Safe Beverage Container as Displayed in Project Website

Dym comparison. The numerical difference doesn't show up in the most important item, because both Dym and Little and the project assign a 1 to the more important objective. The difference is due to Dym assigning a 0 to the less important objective and the project site assigning a -1. For example, in Figure 5.5 the BJIC weighted objective, "Appeals to Parents" has three votes for, and two votes against giving it a score of 3. In Figure 5.6, "Appeals to parents" has two votes for going across the row, and one vote for and two votes against down the column. This gives tallies of 3 and -2 resulting in a total of 1. Regardless of the totals, the rank order is the same in the two figures. The Dym and Little method of displaying and tallying votes is more intuitive. No modifications to the underlying database would be necessary to accommodate an alternative display style, but modifications of the pairwise.php page would be necessary.

The Dym and Little textbook does cover identifying specifications and metrics, but it does not discuss these topics with regards to the safe beverage container.

Because most users are familiar with some beverage container designs, the next step of defining functions, is done purely as a listing. The functions served by the beverage container as listed in **Engineering Design** include

- contain liquid
- get liquid into the container (fill the container)
- get liquid out of the container (empty the container)
- close the container after opening (if it is to be used more than once)
- resist forces induced by temperature extremes
- resist forces induced by handling in transit
- identify the product

These same functions can be written to the `leaf` table using the use.php script (Figure 5.7). These function leaves are associated with the "Safe Beverage Container Functions" branch.

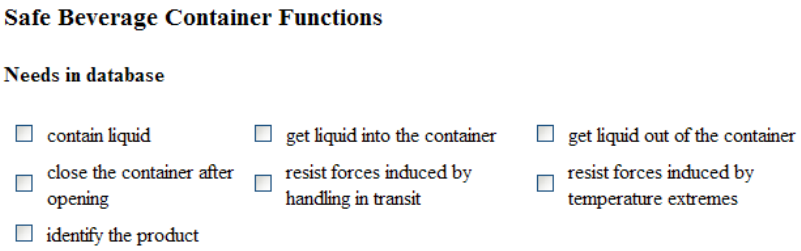


Figure 5.7: Functions for the Safe Beverage Container as Displayed in Project Website

After the functions are determined, the concepts are developed. These concepts are entered through use.php and stored in the `leaf` table under the "Safe Beverage Container Concepts" branch. After the concepts are developed, they are compiled into a morphological chart. The morphological chart displayed in **Engineering Design** (Figure 5.8) is based off some of the functions previously listed, but there has been some modification in the functions. The morphological chart shown from morphchart.php (Figure 5.9) has

MEANS FEATURE/FUNCTION	1	2	3	4	5	6
Contain beverage	can	bottle	bag	box	••••	••••
Material for drink container	aluminum	plastic	glass	waxed cardboard	lined cardboard	mylar films
Mechanism to provide access to juice	pull tab	inserted straw	twist top	tear corner	unfold container	••••
Display of product information	shape of container	labels	color of material	••••	••••	••••
Sequence manufacture of juice container	concurrent	serial	••••	••••	••••	••••

Figure 5.8: Morphological Chart From **Engineering Design** by Dym and Little [3][p 106]

modified the functions from Figure 5.7 to reflect those used in Figure 5.8 The axes on the morphological charts are switched, but the information contained within the chart is identical.

The safe beverage container was reasonably simple to conduct using the existing user interface. The lack of specifications presented did not affect the population of the rest of

Contain beverage	Material for drink container	Mechanism to provide access to juice	Display of product information	Sequence manufacture of juice container
Can	glass	Pull tab	Shape of container	concurrent
Bottle	waxed cardboard	Inserted straw	Labels	serial
Bag	aluminum	Twist top	Color of material	
Box	lined cardboard	Tear corner		
	plastic	Unfold container		
	mylar films			

Figure 5.9: Morphological Chart From Project Website.

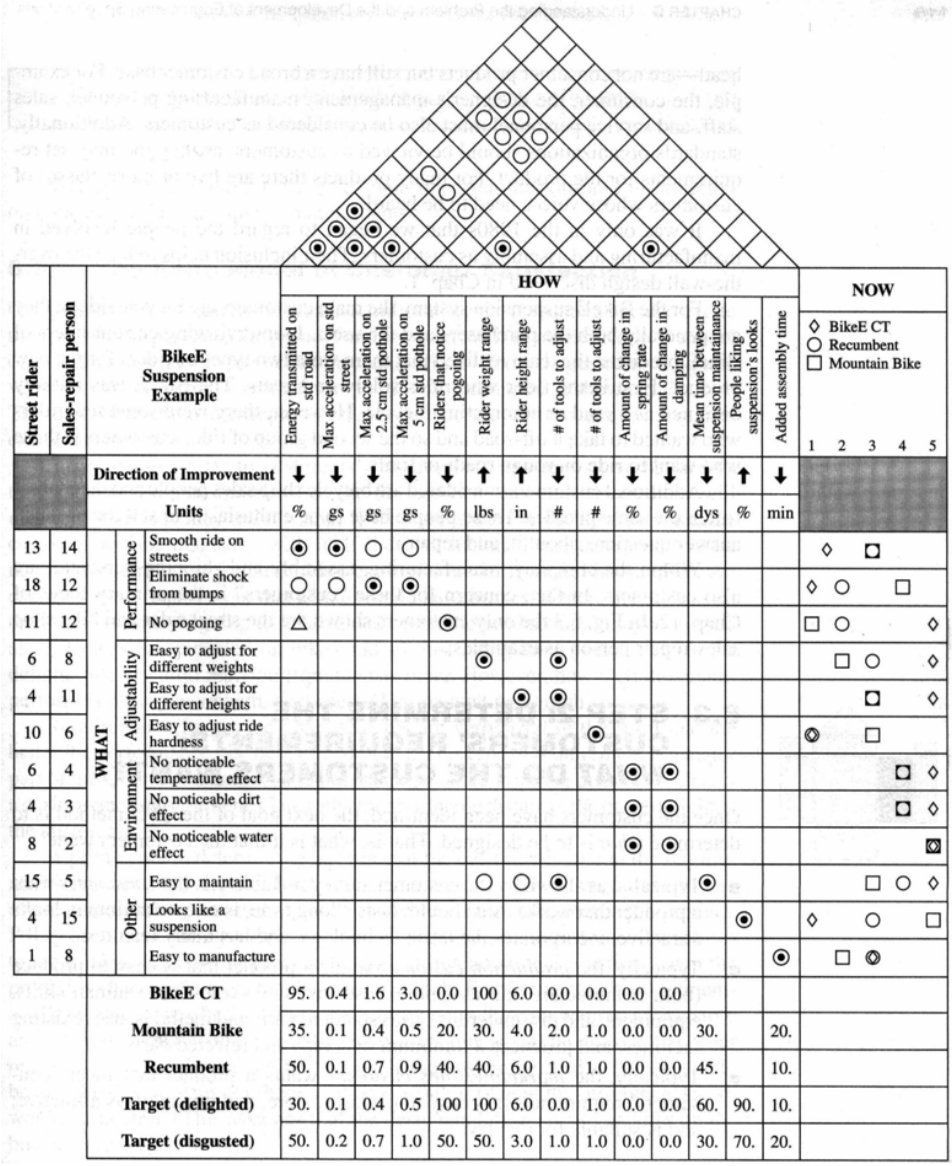
the information. Graphical displays were generally similar and recognizable.

5.2 BikeE

The second case study is an example from **The Mechanical Design Process** by David Ullman [9]. The case study used in Ullman follows the development of a bicycle rear suspension system. Ullman uses a House of Quality (Figure 5.10) to display the information gathered in the beginning stages of design. The project website does not directly display information in the form of a completed House of Quality, yet most of the information contained can be displayed through the tools provided in the web site.

The BikeE suspension customer needs were determined based on customer surveys and compiled into a list. The customer requirements from the book are shown in (Figure 5.11). These customer requirements are entered using use.php and stored in the `leaf` table under the branch "BikeE Customer Needs". The project displays four requirements that aren't on the book list (Figure 5.12). This is due to the four requirement groupings shown on the HoQ. While Ullman doesn't address using objective trees or affinity diagrams, the House of Quality does show the requirements sorted under the What heading.

Before covering the requirements, Ullman identifies who the users are. The project web site does not currently have a functionality to determine users. After establishing the different end users, the users are asked to weight the requirements. The method used is an alternative to some of the more traditional weighting methods. The user is given 100 points and is asked to distribute the points between the requirements, where more points



Street rider	Sales-repair person			BikeE Suspension Example	Energy transmitted on std road	Max acceleration on std street	Max acceleration on 2.5 cm std pothole	Max acceleration on 5 cm std pothole	Riders that notice pogoing	Rider weight range	Rider height range	# of tools to adjust	# of tools to adjust	Amount of change in spring rate	Amount of change in damping	Mean time between suspension maintainance	People liking suspension's looks	Added assembly time	NOW ◇ BikeE CT ○ Recumbent □ Mountain Bike 1 2 3 4 5
				Direction of Improvement	↓	↓	↓	↓	↓	↑	↑	↓	↓	↓	↓	↓	↑	↓	
				Units	%	gs	gs	gs	%	lbs	in	#	#	%	%	dys	%	min	
13	14	WHAT	Performance	Smooth ride on streets	◉	◉	○	○											◇ □
18	12	WHAT	Performance	Eliminate shock from bumps	○	○	◉	◉											◇ ○ □
11	12	WHAT	Performance	No pogoing	△				◉										□ ○ ◇
6	8	WHAT	Adjustability	Easy to adjust for different weights						◉		◉							□ ○ ◇
4	11	WHAT	Adjustability	Easy to adjust for different heights							◉	◉							□ ◇
10	6	WHAT	Adjustability	Easy to adjust ride hardness									◉						◈ □
6	4	WHAT	Environment	No noticeable temperature effect										◉	◉				□ ◇
4	3	WHAT	Environment	No noticeable dirt effect										◉	◉				□ ◇
8	2	WHAT	Environment	No noticeable water effect										◉	◉				◈
15	5	WHAT	Other	Easy to maintain						○	○	◉				◉			□ ○ ◇
4	15	WHAT	Other	Looks like a suspension													◉		◇ ○ □
1	8	WHAT	Other	Easy to manufacture														◉	□ ◈
				BikeE CT	95.	0.4	1.6	3.0	0.0	100	6.0	0.0	0.0	0.0	0.0				
				Mountain Bike	35.	0.1	0.4	0.5	20.	30.	4.0	2.0	1.0	0.0	0.0	30.		20.	
				Recumbent	50.	0.1	0.7	0.9	40.	40.	6.0	1.0	1.0	0.0	0.0	45.		10.	
				Target (delighted)	30.	0.1	0.4	0.5	100	100	6.0	0.0	1.0	0.0	0.0	60.	90.	10.	
				Target (disgusted)	50.	0.2	0.7	1.0	50.	50.	3.0	1.0	1.0	0.0	0.0	30.	70.	20.	

Figure 5.10: House of Quality From **The Mechanical Design Process** [9][p 117]

Smooth ride on streets
Eliminate shocks from bumps
Easy to adjust suspension system for different weight riders
Easy to adjust suspension system for different height riders
Easy to adjust suspension system for ride hardness
Easy to maintain
Looks like a suspension
Not noticeably affected by temperature
Not noticeably affected by dirt
Not noticeably affected by water
No pogoing[1]
Easy to assemble
Cost less than $50 to manufacture over the rigid rear fork
Weigh less than 400 grams over the rigid rear fork
Does not change bike height

[1] A bike that pogos moves up each time a pedal is pushed so it bounces up and down twice for each pedal revolution. Pogoing results from a poor design where the chain tension interacts with the suspension.

Figure 5.11: Customer Requirements From **The Mechanical Design Process** [9][p 127]

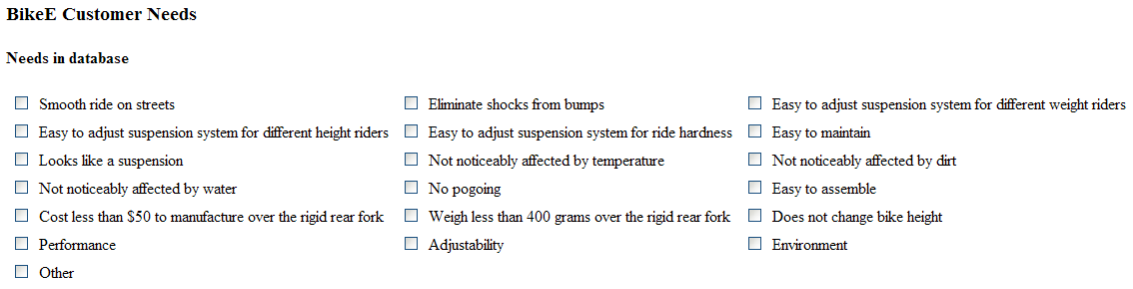


Figure 5.12: Customer Needs From Project Website.

are given to the needs with the greater importance. The tool offered in the project website for assigning importance is the pairwise comparison tool, but a more appropriate method for this distributive weighting would be to implement a tool to allow a team member in a role similar to the "chief engineer" at Toyota to assign an importance rank.

After determining the Who vs. What portion of the HoQ, the next step is to establish a base line of Now vs. What. This is the measure of effectiveness. The Now vs. What compares how well the current options satisfy the customer. This portion is as of yet unaddressed by the project software and it is an area that could be used to implement the Reverse Engineering process documented in Frank Tamarez Gomez' thesis [38].

The How portion of the HoQ documents the engineering specifications and lists them across the top. The specifications are shown as displayed by use.php in Figure 5.13. These specifications are entered in as leaves under the "BikeE Specifications" branch. The How portion also includes information on units of the specifications and the direction of improvement. For example, the first specification - "energy transmitted on standard road" - is measured in a percent, and will be better if that percentage is reduced.

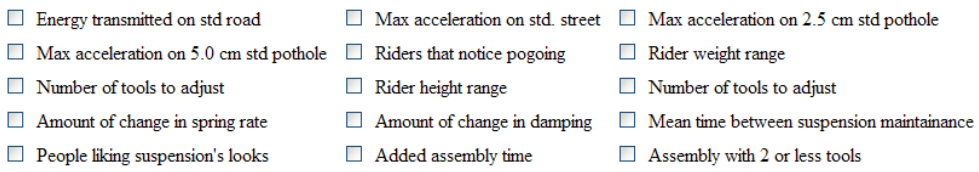


Figure 5.13: Engineering Specifications From Project Website.

The main part of the HoQ is showing how the customer requirements translate into engineering specifications. The different icons in in the what vs. how grid display the strength of relationship. When establishing relationships between customer needs and engineering specifications through use.php, the information written to the `beta` table only indicates that a relationship exists, not the strength of the relationship. The `beta` table allows for indications of the strength of relationship through the `data` field, although the use.php doesn't have a method to enter in the information. Figure 5.14 shows comparison.php

rendering the comparison with the strength of relationship included, which was manually entered into `beta`. The 9 stands for a strong positive relationship and represents the ◎. The 3 stands for a positive relationship and represents the ◯. The 1 stands for a weak positive relationship and represents the △.

	Energy transmitted on std road	Max acceleration on std. street	Max acceleration on 2.5 cm std pothole	Max acceleration on 5.0 cm std pothole	Riders that notice pogoing	Rider weight range	Number of tools to adjust	Rider height range	Number of tools to adjust	Amount of change in spring rate	Amount of change in damping	Mean time between suspension maintainance	People liking suspension's looks	Added assembly time
Smooth ride on streets	9	9	3	3										
Eliminate shocks from bumps	3	3	9	9										
Easy to adjust suspension system for different weight riders						9	9							
Easy to adjust suspension system for different height riders							9	9						
Easy to adjust suspension system for ride hardness							9							
Easy to maintain						3	9	3				9		
Looks like a suspension													9	
Not noticeably affected by temperature										9	9			
Not noticeably affected by dirt										9	9			
Not noticeably affected by water										9	9			
No pogoing	1				9									
Easy to assemble														9

Figure 5.14: Customer Requirements vs. Engineering Specifications From Project Website.

The basement of the HoQ has the measures of performance, including a minimum level of satisfaction and a maximum level of satisfaction, as well has how well each product satisfies the performance.

The roof of the HoQ identifies how specifications are dependent on each other. Some specifications have a positive influence on each other, while others have negative influence. In the case of negative influence, it's necessary to make some tradeoff decisions. The website is set up to store the information in `beta` and display a tradeoff matrix through the script comparison.php, but it's not set up to accept the user input. By entering in the information contained in the roof directly into the `beta` table, a rudimentary roof can be displayed with comparison.php. Figure 5.15 shows the comparison table with specifications against specifications. This square matrix shows the information twice. The gray has been added afterwards to reinforce the information necessary. The 9 stands for a strong positive relationship and represents the ◎. The 3 stands for a positive correlation and represents the ◯. Negative and strong negative relationships, while not used in the

case study example, are also possible correlations that would be stored as -1 and -3 respectively.

	Energy transmitted on std road	Max acceleration on std. street	Max acceleration on 2.5 cm std pothole	Max acceleration on 5.0 cm std pothole	Riders that notice pogoing	Rider weight range	Number of tools to adjust	Rider height range	Number of tools to adjust	Amount of change in spring rate	Amount of change in damping	Mean time between suspension maintainance	People liking suspension's looks	Added assembly time
Energy transmitted on std road		9	9	9										
Max acceleration on std. street	9		9	9				3		3	3			
Max acceleration on 2.5 cm std pothole	9	9		9				3		3	3			
Max acceleration on 5.0 cm std pothole	9	9	9					3		3	3			
Riders that notice pogoing														
Rider weight range								9						
Number of tools to adjust														
Rider height range		3	3	3		9								
Number of tools to adjust														
Amount of change in spring rate		3	3	3							9			
Amount of change in damping		3	3	3						9				
Mean time between suspension maintainance														
People liking suspension's looks														
Added assembly time														

Figure 5.15: House of Quality Shaded Roof From Project Website.

In entering in the information to be displayed for the roof of the House of Quality, an area of concern was recognized. By including the explicit relationship between specifications, the comparison chart displayed a trickle down effect creating implicit relationships between customer needs and engineering specifications that aren't identified in the original House of Quality. Inspection of other textbooks [13, 14, 16] confirmed that in fact several HoQ examples do not recognize this implicit relationship. Referring back to Figure 5.10, the customer need "smooth ride on streets" shows a strong relationship with "max acceleration on standard street". "Max acceleration on standard street" in turn shows a positive correlation through the roof with "amount of change in spring rate". There is no indication of any relationship between "smooth ride on streets" and "amount of change in spring rate". The implications of this implicit relationship call into question the validity of the house of quality. Further exploration is beyond the scope of this thesis.

Using the Ullman case study of the House of Quality is a great road map for further paths of development for Needs Assessment. Offering all of the functionality that goes into completely filling out (Figure 5.10) would provide complete coverage of the Needs

Assessment determination.

For concept development, Ullman displays a morphological chart to compare project functions with the concepts developed to satisfy those functions.(Figure 5.16) While the axis are switched, the same functionality is achieved in (Figure 5.17). Both the functions and concepts are written to the database as leaves through use.php. The relationship between functions and concepts are stored in the `beta` table. The morphological chart is rendered using morphchart.php.

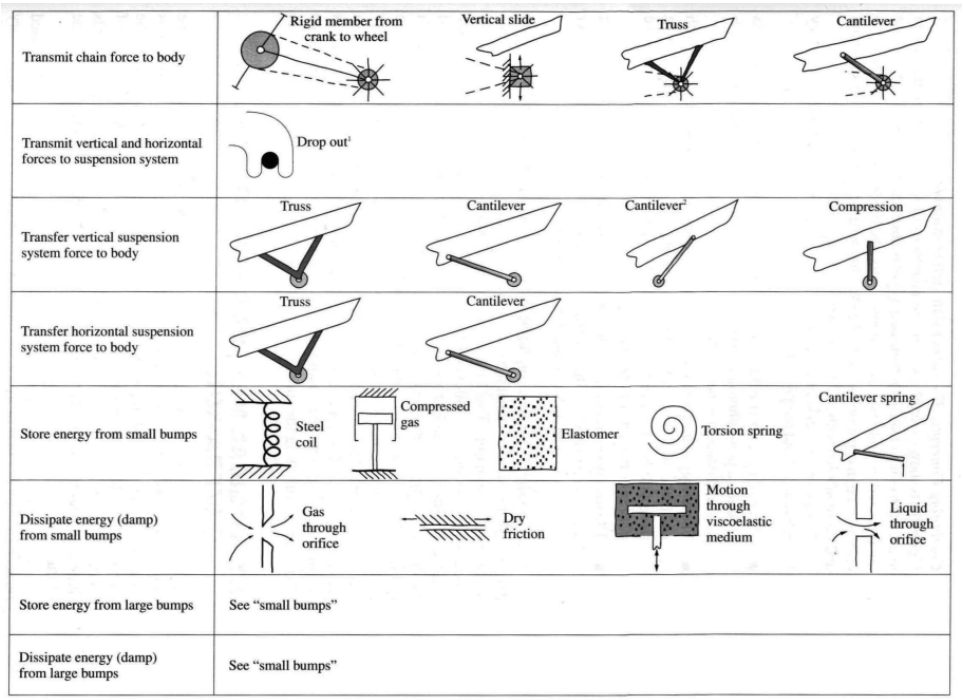


Figure 5.16: Morphological Chart From **The Mechanical Design Process** [9][p 165]

The BikeE case study demonstrates the need to provide alternative methods for displaying data. It also highlights areas of the project that are as of yet uncovered.

Transmit chain force *to body*	Transmit vertical and horizontal forces to suspension system	Transfer vertical suspension system force to body	Transfer horizontal suspension system force to body	Store energy from small bumps	Dissipate energy (damp) from small bumps	Store energy from large bumps	Dissipate energy (damp) from large bumps
Rigid member from crank to wheel	Drop out	Cantilever	Cantilever	Steel coil	Gas through orifice	Steel coil	Gas through orifice
Vertical slide		Truss	Truss	Compressed gas	Dry friction	Compressed gas	Dry friction
Truss		Inverse cantilever		Elastomer	Motion through viscoelastic medium	Elastomer	Motion through viscoelastic medium
Cantilever		Compression		Torsion spring	Liquid through orifice	Torsion spring	Liquid through orifice
				Cantilever spring		Cantilever spring	

Figure 5.17: Morphological Chart From Project Website.

5.3 Alternative Views

The value to be gained in this project is the ability to take information entered using tools in one process view, and display it in different tools in another process view. For example, does the information contained in the BikeE case study make sense when displayed in the Dym and Little process view style? Likewise, does the Safe Beverage Container information translate into the Ullman process view style.

Dym and Little do not discuss the House of Quality in **Engineering Design** although it dominates needs assessment in Ullman's text. In order to present the Safe Beverage Container in a view that's familiar to a user of the Ullman process, it should be presented using the comparison chart. This becomes an issue due to the lack of specification laid out by Dym. One look at the Safe Beverage Container comparison chart (Figure 5.18) by a team member using the Ullman design process and it's instantly clear that a major component of the design process has not yet been addressed. Likewise, if information on the BikeE project needs to be shared with a design team that is more familiar with the Dym and Little process, that information should be able to provided in the familiar format. The Ullman HoQ showed needs grouped into sections, but that may be better view to a Dym and Little user as an objective tree, as shown in Figure 5.19.

The pairwise comparison is another tool familiar to Dym and Little users, but not Ullman users. Based on the weighted importance rankings shown in Figure 5.10, the relative importance was entered into a pairwise comparison (Figure 5.20). The benefit of displaying the weighted rankings in a pairwise comparison matrix is the ability to verify that the

Safe
Environmentally benign
Promotes sales
Easy to distribute
Preserves taste
Appeals to parents
Permits marketing flexibility
Generates brand identity
Durable for shipment
Easy for kids to use
Perceived as safe
Inexpensive to produce
Distinctive appearance
Resists Forces and Shocks
Resists Range of Temperatures
Easy to open
Hard to spill

Figure 5.18: Safe Beverage Container Comparison Chart From Project Website.

end results make sense. Due to every need being shown against every other need, it should confirm the rank order of importance.

5.4 Case Study Summary

Based on the case studies, it is evident that the database is appropriate for storing the qualitative data for the scripts in place. It is also evident that there is significant room for further development in the tools made available to the user. The holes in the case studies were due to lack of information provided by the text, such as specifications for the safe beverage container. The HoQ by Ullman is an excellent guideline of additional functionality that should be investigated to completely capture the needs assessment, for example, identifying multiple users, benchmarking, or applying metrics.

BikeE Customer Needs

- ○ Performance
  - ○ Smooth ride on streets
  - ○ Eliminate shocks from bumps
  - ○ No pogoing
- ○ Adjustability
  - ○ Easy to adjust suspension system for different weight riders
  - ○ Easy to adjust suspension system for different height riders
  - ○ Easy to adjust suspension system for ride hardness
- ○ Environment
  - ○ Not noticeably affected by temperature
  - ○ Not noticeably affected by dirt
  - ○ Not noticeably affected by water
- ○ Other
  - ○ Easy to maintain
  - ○ Looks like a suspension
  - ○ Easy to assemble

Figure 5.19: BikeE Objective Tree From Project Website.

	Smooth ride on streets	Eliminate shocks from bumps	No pogoing	Easy to adjust suspension system for different weight riders	Easy to adjust suspension system for different height riders	Easy to adjust suspension system for ride hardness	Not noticeably affected by temperature	Not noticeably affected by dirt	Not noticeably affected by water	Easy to maintain	Looks like a suspension	Easy to assemble
Smooth ride on streets		-1	1	1	1	1	1	1	1	-1	1	1
Eliminate shocks from bumps			1	1	1	1	1	1	1	1	1	1
No pogoing				1	1	0	1	1	1	-1	1	1
Easy to adjust suspension system for different weight riders					1	-1	0	1	-1	-1	1	1
Easy to adjust suspension system for different height riders						-1	-1	0	-1	-1	0	-1
Easy to adjust suspension system for ride hardness							1	1	1	-1	1	1
Not noticeably affected by temperature								1	-1	-1	1	1
Not noticeably affected by dirt									-1	-1	0	1
Not noticeably affected by water										-1	1	1
Easy to maintain											1	1
Looks like a suspension												1
TOTALS	7	11	4	-2	-9	4	-2	-7	1	9	-7	-9

Figure 5.20: BikeE Pairwise Comparison From Project Website.

The flow of information is demonstrated in the morphological charts (Figures 5.9 and 5.17) and the comparison chart (Figure 5.14). These display how functions drive brainstorming concepts and how specifications address needs. The flow of information is also demonstrated in showing how information displayed using one rendering tool for one user, can be shown in a different rendering tool for another user.

There's an additional capability of the scripts that was not displayed through the case studies. This is the implicit relationships capable of being displayed. This was not demonstrated in the case study because current literature, Dym and Little and Ullman included, do not map how specifications relate to functions. To demonstrate proof of concept for this thesis, relationships were manually added to the table `beta` relating several BikeE engineering specifications to design functions. Given the already stated needs to specifications relationships, and the already stated functions to concepts relationships, with the addition of the specifications to functions it becomes feasible to display implicit relationships between any of these four branches, including concepts to customer needs as shown in Figure 5.21. By showing the connections between the customer needs and the design concepts, information flows between "Needs Assessment" and "Concept Development". It ensures that the design team is keeping the customer needs in mind throughout the process. No concepts should be offered that do not address a need and no stated needs should be without a concepts to deal with them.

	Rigid member from crank to wheel	Vertical slide	Truss	Cantilever	Drop out	Inverse cantilever	Compression	Steel coil	Compressed gas	Elastomer	Torsion spring	Cantilever spring	Gas through orifice	Dry friction	Motion through viscoelastic medium	Liquid through orifice
Smooth ride on streets		X		X	X	X	X									
Eliminate shocks from bumps		X		X	X	X	X									
Easy to adjust suspension system for different weight riders																
Easy to adjust suspension system for different height riders																
Easy to adjust suspension system for ride hardness																
Easy to maintain																
Looks like a suspension																
Not noticeably affected by temperature													X	X	X	X
Not noticeably affected by dirt													X	X	X	X
Not noticeably affected by water													X	X	X	X
No pogoing			X	X	X	X	X									
Easy to assemble																
Cost less than $50 to manufacture over the rigid rear fork																
Weigh less than 400 grams over the rigid rear fork																
Does not change bike height																
Performance																
Adjustability																
Environment																
Other																

Figure 5.21: BikeE Comparison Between Customer Needs and Suspension Design Concepts From Project Website.

Chapter 6

Conclusions

A database-driven series of web-based scripts were developed, implemented, and evaluated using case studies to facilitate information flow between tools early in the engineering design process. The following goals were laid out in the Statement of Work in Chapter 1:

- Determine similar phases early in engineering design process.
- Identify common tools used in the early facets of the design process.
- Propose a table structure to store information from identified tools.
- Implement the table structure through a series of web-based scripts.
- Demonstrate the web-based scripts through case studies.
- Evaluate the table structure and implementation.
- Recommend considerations for future work.

All stated goals have been met. Customer needs assessment and concept development were identified to be routinely implemented early in existing engineering design processes in Section 2.2.

Commonly utilized tools in these early facets were found in Section 2.3 to include: brain ball, brainstorming, and collaborative sketching as methods of generating raw ideas, affinity diagrams, functions-means trees, and objective trees as methods of grouping similar ideas together, pairwise comparisons and weighted voting as methods for creating a

ranking of importance or appropriateness, and House of Quality and morphological charts as methods of representing relationships between information.

A schema consisting of eight tables was proposed in Section 4.2 to store the information associated generating raw ideas, grouping similar ideas together, ranking the importance or appropriateness of the information, and storing relationships between the information.

A series of scripts discussed in Section 4.3 were written in php and implemented on an apache web site, using a MySQL database to provide a user interface to the schema. The pages provided included welcome, existing, use, pairwise, comparison, morphchart and voting. These pages wrote the information to the schema and rendered the information back to the user. Comparison displayed the graphical representation of the information flow between tools.

Selected case studies documented in Chapter 5 were conducted based on examples given in **Engineering Design** by Dym and Little and **The Mechanical Design Process** by Ullman. The case studies identified the strength of the database in supporting the information as well as shortcomings in the user interface as well as the text examples. The schema structure supported the case studies in instances where the user interface could not. The case studies highlighted the ability of information flow between design tools utilized by different processes. The case study also demonstrated the flow of information between "Needs Assessment" and "Concept Development" through a comparison chart displaying the implicit relationships between customer needs and design concepts.

Through the conducted research and the case study, two areas of concern were identified; (1) the disconnect between engineering specification definition and function definition and (2) the apparent inconsistencies of implicit relationships within the House of Quality. Further research beyond the scope of this thesis is recommended to identify and understand the relationships and information flow between engineering specifications and product functions as the texts reviewed do not address this area. Additionally, further investigation into the nature of the implicit relationships within the House of Qualify is advised. As shown in the Chapter 5, implicit relationships were identified between customer needs

and engineering specifications that were not explicitly stated.

Additional recommendations for further work in this area beyond these two areas of concern are presented in Chapter 7. There are many potential benefits to the user to be achieved once the recommendations are implemented. It is strongly urged that development on this thesis project continues.

Chapter 7

Recommendations

In addition to the recommendations made in the conclusions, a large number of areas for improvement were identified based on the initial development of the thesis project. These recommendations for further work are grouped into three categories: immediate need for implementation, future development opportunities, and a list of suggestions for improvement.

7.1 Immediate Needs

- Implement into EDGE environment through installing database and scripts on the edge server. Database name, user name and password modifications will be necessary.
- Develop wiki-based user interface. Modify scripts to function similar to the scripts already used in EDGE.
- Improve ease of brainstorming concepts by selecting a single function, having that selected function stay "on" and only displaying that functions' concepts. The current method is bulky and non-intuitive.
- Add ability to assign multiple parents to a leaf in the sorting view of use.php, perhaps by using a checkbox rather than a select circle.

- Display confirmation before deleting a leaf.
- Differentiate between display of implicit relationships and explicit relationships in comparison.php by using different symbols. Explicit relationships are entered in the `beta` table while implicit relationships are derived from other explicit relationships.
- Determine appropriate method of storing strength of relationship data (`beta.data` vs. `gamma.data`). See page 69 for further information.

7.2 Future Development Opportunities

- Develop a set of graphical renders to better display information.
- Incorporate the ability to consolidate input from multiple tools such as a brainstorming session and a brainball session or an interview and a focus group. It is desirable to keep the history of where information came from, but be able to deal with leaves on separate branches as if they were on a single branch.
- Update comparison.php so cells can be edited to write relationships to `beta`. Clicking on a cell and directly manipulating the relationships would be a much simpler interface. It would also allow for varying values.
- Develop a tool to allow a "chief engineer" style weighted importance ranking to be written to `alpha`.
- Display items in morphchart.php in rank order. Include links to concept generation or voting. For example, clicking on a function column header where there are no existing concepts would allow the user to generate concepts for that function or if there were existing concepts, then the user could vote on the appropriateness of those concepts.

- Identify the customers for the project and relate the importance of the customer need relative to the different end users. In other words, allow several entries in the `alpha` table, depending on the end customer.
- Offer ability to add images such as used in the c-sketch method. This could be simply a link to a location where the drawing is stored instead of a descriptive line in the `leaf` table.
- Create appropriate table(s) and user interface for metrics to add capability for quantifying data. This metric table should include a description of the metric, a foreign key to the leaf, a min, a max and a target value as well as the `id`, `TimeStamp` and `User` fields.
- Expand tools set to the additional 10 facets of EDGE.
- Make recommendations to the user based on information stored in the database. This is more appropriate further down the road, as the size and contents of the database expand.
- Evaluate the effectiveness of performing an analytical analysis on validity of pairwise comparison [26].
- Offer user the opportunity to evaluate the usefulness of the information displayed, similar to Amazon.com reviews.

7.3 Suggested Improvements

- Adding the needs assessment one line at a time is tedious. A better placing template for inputting bulk information would be an improvement (maybe comma separated values?)
- `Leaf.longer` currently has a length restriction. Evaluate the negative or positive values of this.

- Implementing a method to expand or contract objective tree levels would improve displaying content.
- When translating a customer need to a specification, there should be a method to identify a need as a constraint.
- Entering in leaves requires the user to click on the entry line every time. The ability to automatically type in more would be more convenient.
- Add options for strength of comparison. — Option A is significantly more critical — Option A is slightly more critical — Equally critical — Option B is slightly more critical — Option B is significantly more critical.
- Within the weighted voting script, have team leader determine the number of votes to assign, when the voting is available, and when to end voting. Enforce the number of votes available to each user.
- The voting results should only be made available to the team leader and faculty.

Glossary

A

ABET acronym for Accreditation Board of Engineering and Technology.

Affinity Diagram a way to organize facts, opinions, ideas and issues into natural groupings as an aid to diagnosis on a complex problem.

alpha a table in the `test` schema that stores data regarding the relative importance of leaves on a single branch.

Apache a freely-available source code implementation of an HTTP (Web) server.

B

beta a table in the `test` schema that stores data regarding the relationships between two leaves.

Bill of Materials a list of all parts in an assembly.

BJIC a fictitious company used in the "Safe Beverage Container" case study.

Brainstorm a method of shared problem solving in which all members of a group spontaneously contribute ideas.

Branch project term for a session of information creation and a table in the `test` schema that store session names and types.

branchfamily a table in the `test` schema that stores the flow of information between branches.

branchtypes a table in the `test` schema that stores the different branch types and assigns a unique id.

C

CAD acronym for Computer Aided Design.

CADD acronym for Computer Aided Drafting and Design.

Concurrent Engineering a management/operational approach which aims to improve product design, production, operation, and maintenance by developing environments in which personnel from all disciplines (design, marketing, production engineering, process planning, and support) work together and share data throughout all phases of the product life cycle.

Cornerstone Design a course at RIT. This course gives students an opportunity to apply foundation courses in mechanical engineering to the solution of an open-ended design problem. Students will learn about project definition, concept development, feasibility assessment, managing design parameter tradeoffs using engineering analysis, and developing a preliminary design drawing package. Teams of students will develop their concept through the stage of working drawings, based on the ANSI standard for Geometric Dimensioning and Tolerancing. The course is intended to prepare students for future ME and multi-disciplinary design courses. [RIT 2006-2007 Undergraduate Bulletin].

CREST A mnemonic for the feasibility analysis of Constraints, Resources, Economics, Scope, Technology.

CSS acronym for Cascading Style Sheets, a simple mechanism for adding style (*e.g.* fonts, colors, spacing) to web documents.

CSV Comma separated value file extension.

D

Database Server a computer that stores a centrally located organized body of related information to be used by network users.

Design Project Management (DPM) a course at RIT. This course focuses on preparing students to take on a leadership role in design project teams. Topics include product development processes, management of design project teams, developing a business case for design projects, understanding customer needs and

translating them into engineering specifications, tools for developing design concepts, tools for assessing the feasibility of design concepts, conducting engineering tradeoffs and analysis to synthesize a preliminary design. Students use the concepts and tools discussed throughout the course in a team-based environment to develop project readiness packages for subsequent use by senior design teams. [RIT 2006-2007 Undergraduate Bulletin].

DFx refers to Design For all desirable attributes.

E

ECAD acronym for Electronic Computer Aided Design.

EDGE acronym for Engineering Design Guide and Environment, it is an open source integrated design environment to foster collaboration within design project teams, and across design teams working on families of closely related projects.

Edge Schema when displayed as `edge`, the name of a schema that contains data on people and projects.

Engineering Design a systematic, intelligent process in which designers generate, evaluate, and specify concepts for devices, systems, or processes whose form and function achieve clients objectives or users needs while satisfying a specified set of constraints.

Engineering Specifications the restatement of the design problem in terms of parameters that can be measured and have target values.

Extensible describes something in information technology such as a program, programming language, or protocol, that is designed so that users or developers can expand or add to its capabilities.

F

FACETS a twelve-faceted design process proposed by Hensel.

Facets Schema when displayed as `facets`, the name of a schema that contains design process data.

Fields the columns of data in a database.

Function-Means Tree a graphical representation of a design's basic and secondary functions.

G

gamma a table in the `test` schema that stores data regarding the appropriateness of a leaf with regards to another leaf.

H

House of Quality a graphical tool that translates customer requirements, based on marketing research and benchmarking data, into an appropriate number of engineering targets to be met by a new product design.

K

KGCOE acronym for Kate Gleason College of Engineering, one of eight colleges within RIT.

L

LDAP acronym for Lightweight Directory Access Protocol, an Internet protocol that email and other programs use to look up information from a server.

Leaf project term for individual pieces of information and a table in the `test` schema that stores pieces of information.

leaffamily a table in the `test` schema that stores relationships between leaves resulting from a grouping such as an objective tree.

M

Morphological Chart a graphical method of displaying concepts against functions.

MS Access a Microsoft database product.

Multi-disciplinary Senior Design (MDSD) a two course sequence at RIT. Students work in design teams in an environment approximating an industrial setting.

Emphasis is placed on teamwork and on developing good oral, written and interpersonal communication skills. In Senior Design I, student teams develop their proposed final design of a mechanical system after identifying possible alternative concepts. The final design must be supported by sound engineering analyses and by engineering drawings necessary to build a prototype. In Senior Design II student teams build and test a working prototype of their previously developed final design. Non-working prototypes are not acceptable, and some redesign work may be required to make the system work. Continued emphasis is placed on teamwork and on developing good oral, written and interpersonal communication skills. [RIT 2006-2007 Undergraduate Bulletin].

MySQL an open source relational database management system (RDBMS) that uses Structured Query Language (SQL) for adding, accessing, and processing data in a database.

N

Normalization the process of simplifying the structure of data [37].

O

Objective the desired attributes of the design.

Objective Tree ordered list of the desired attributes of a design.

Open Architecture a type of computer architecture or software architecture that allows adding, upgrading and swapping components.

Open Source computer software whose source code is available under a license (or arrangement such as the public domain) that permits users to use, change, and improve the software, and to redistribute it in modified or unmodified form.

P

Pairwise Comparison tool used to help understand the relative importance of items being compared.

PHP acronym for Hypertext Preprocessor, it is a programming language that allows web developers to create dynamic content that interacts with databases for web based software applications.

Prototype a facsimile of an end product used to demonstrate a concept rapidly, check feasibility, and/or gain acceptance.

Q

Quality Function Deployment an overall methodology that begins in the design process and attempts to map the customer-defined expectation and definition of quality into the processes and parameters that will fulfill them.

R

RDBMS acronym for Relational Database Management System, a database management system in which the database is organized and accessed according to the relationships between data values.

Record the row of information in a database.

RIT Rochester Institute of Technology.

S

Schema a synonym for database, a relational collection of tables.

Script a program that may accompany an HTML document or be embedded directly in it.

SQL acronym for Structured Query Language, it is a language that provides an interface to relational database systems.

Subversion (SVN) an open source version control system.

T

Table a set of data arranged in rows and columns.

Test Schema when displayed as `test`, the name of a schema that contains project data.

Tree project term for the project analogy.

W

Wiki a web application designed to allow multiple authors to add, remove, and edit content.

Bibliography

[1] M. Asimow. *An Introduction to Design*. Prentice-Hall, Englewood Cliffs, NJ, 1962.

[2] K.T. Ulrich and S.D. Eppinger. *Product Design and Development*. McGraw-Hill/Irwin, New York, 2004.

[3] C.L. Dym and P. Little. *Engineering Design, a project based introduction*. John Wiley & Sons, Inc., 2nd edition, 2004.

[4] A. Ertas and J. Jones. *The Engineering Design Process*. John Wiley & Sons, Inc., New York, 2nd edition, 1996.

[5] R. Ford and C. Coulston. *Design for Electrical and Computer Engineers*. McGraw Hill, Boston, 2005.

[6] R. Cooper. Stage-gate systems: a new tool for managing new products. *Business Horizons*, pages 44 – 53, 1990.

[7] R. Phillips, K. Neailey, and T. Broughton. Comparative study of six stage-gate approaches to product development. *Integrated manufacturing systems*, 10(5):289 – 297, 1999.

[8] M. Fleischer and J. Liker. *Concurrent Engineering Effectiveness: Integrating Product Development Across Organizations*. Hanser Gardner Publications, Cincinatti, 1997.

[9] D.J. Ullman. *The Mechanical Design Process*. McGraw-Hill, New York, 3rd edition, 2003.

[10] Engineering Accreditation Commission, Baltimore, MD. *Criteria for Accrediting Engineering Programs*, October 2005.

[11] C.L. Dym, A.M. Agogino, O. Eris, D.D. Frey, and L.J. Leifer. Engineering design thinking, teaching, and learning. *Journal of Engineering Education*, 94:103–119, 2005.

[12] A. Griffin. Pdma research on new product development practices: Updating trends and benchmarking best practices. *Journal of Product Innovation Management*, 14:429 – 458, 1997.

[13] G. E. Dieter. *Engineering Design*. McGraw-Hill, Boston, 2000.

[14] K. Otto and K. Wood. *Product Design*. Prentice Hall, Upper Saddle River, NJ, 2001.

[15] M. N. Horenstein. *Design Concepts for Engineers*. Prentice-Hall Engineering Source, Upper Saddle River, NJ, 2006.

[16] N.R. Tague. *The Quality Toolbox*. ASQ Quality Press, Milwaukee, WI, 2nd edition, 2005.

[17] A.F. Osborn. *Your creative power: how to use imagination*. C. Scribner's Sons, New York, 1948.

[18] P. B. Paulus, T. Nakui, and V. L. Putman. Group brainstorming and teamwork: Some rules for the road to innovation. In L. Thompson and H. Choi, editors, *Creativity and innovation in organizational teams*, pages 69–86. Lawrence Erlbaum Associates, 2005.

[19] N. L. Oxley, M. T. Dzindolet, and P. B. Paulus. The effects of facilitators on the performance of brainstorming groups. *Journal of Social Behavior and Personality*, 11:633–646, 1996.

[20] J. E. McGrath. *Groups: Interaction and performance*. Prentice-Hall, Englewood Cliffs, NJ, 1984.

[21] M. Yang and J. Cham. An analysis of sketching skill and its role in early stage engineering design. *Journal of Mechanical Design*, 129:476–482, May 2007.

[22] M. Leary and C. Burvill. Enhancing the quality function deployment conceptual design tool. *Journal of Mechanical Design*, 129(7):701–708, 2007.

[23] N. Cross. *Engineering Design Methods: Strategies for Product Design.* Wiley, New York, 1994.

[24] C.L. Dym, W.H. Wood, and M.J. Scott. Rank ordering engineering designs: Pairwise comparison charts and borda counts. *Research in Engineering Design*, 13:236 – 242, 2002.

[25] T. See and K. Lewis. A formal approach to handling conflicts in multiattribute group decision making. *Journal of Mechanical Design*, 128:678–688, July 2006.

[26] K. Osei-Bryson. An action learning approach for assessing the consistency of pairwise comparison data. *European Journal of Operational Research*, 174(1):234–244, 2006.

[27] E. DeBartolo. Development of an introduction to mechanical engineering design course. In *Proceedings of the 2002 American Society for Engineering Education Annual Conference and Exposition*, Montral, Quebec, Canada, 2002.

[28] E. Hensel and P. Stiebitz. A design project management course at rit. In *Proceedings of the 2003 American Society for Engineering Education Annual Conference and Exposition*, Nashville, TN, 2003.

[29] P. Stiebitz, E. Hensel, and J. Mozrall. Multidisciplinary engineering senior design at rit. In *Proceedings of the 2004 American Society for Engineering Education Annual Conference and Exposition*, Salt Lake City, UT, 2004.

[30] M. Esterman, D. Patru, V. Amuso, E. Hensel, and M. Smith. Development of integrated project tracks for a college-wide multidisciplinary engineering design program at rit. In *Proceedings of the 2007 American Society for Engineering Education Annual Conference and Exposition*, Honolulu, Hawaii, 2007.

[31] H. Palmer. Catalyzing systemic change towards a multidisciplinary, product innovation focus. In *Proceedings of the 2007 American Society for Engineering Education Annual Conference and Exposition*, Honolulu, Hawaii, 2007.

[32] M. Bailey and E. DeBartolo. Using the experiential learning model and course assessment to transform a multidisciplinary senior design course sequence. In *Proceedings of the 2007 American Society for Engineering Education Annual Conference and Exposition*, Honolulu, Hawaii, 2007.

[33] W. Walter, J. Webb, M. Smith, E. DeBartolo, M. Bailey, and G. Slack. Redesigning a college-wide multidisciplinary engineering design program at rit. In *Proceedings of the 2007 American Society for Engineering Education Annual Conference and Exposition*, Honolulu, Hawaii, 2007.

[34] E. Hensel. A multi-faceted design process for multi-disciplinary capstone design projects. In *Proceedings of the 2001 American Society for Engineering Education Annual Conference and Exposition*, Albuquerque, NM, 2001.

[35] TRIZ Institute. The triz journal. `www.triz-journal.com`.

[36] K.E. Nordland. US Patent 7,124,910: Leak-resistant polymeric foam containers, 2006.

[37] G. Powell. *Beginning Database Design and Implementation*. Wiley Publishing, Indianapolis, Indiana, 2006.

[38] F. Tamarez G. A reverse engineering process for mechanical engineering systems. Master's thesis, Rochester Institute of Technology, 2007.